Topology-based Character Motion Synthesis

Shu Lim Ho

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2010

Abstract

This thesis tackles the problem of automatically synthesizing motions of close-character interactions which appear in animations of wrestling and dancing. Designing such motions is a daunting task even for experienced animators as the close contacts between the characters can easily result in collisions or penetrations of the body segments. The main problem lies in the conventional representation of the character states that is based on the joint angles or the joint positions. As the relationships between the body segments are not encoded in such a representation, the path-planning for valid motions to switch from one posture to another requires intense random sampling and collision detection in the state-space.

In order to tackle this problem, we consider to represent the status of the characters using the spatial relationship of the characters. Describing the scene using the spatial relationships can ease users and animators to analyze the scene and synthesize close interactions of characters. We first propose a method to encode the relationship of the body segments by using the Gauss Linking Integral (GLI), which is a value that specifies how much the body segments are winded around each other. We present how it can be applied for content-based retrieval of motion data of close interactions, and also for synthesis of close character interactions. Next, we propose a representation called Interaction Mesh, which is a volumetric mesh composed of points located at the joint position of the characters and vertices of the environment. This raw representation is more general compared to the tangle-based representation as it can describe interactions that do not involve any tangling nor contacts. We describe how it can be applied for motion editing and retargeting of close character interaction while avoiding penetration and pass-throughs of the body segments.

The application of our research is not limited to computer animation but also to robotics, where making robots conduct complex tasks such as tangling, wrapping, holding and knotting are essential to let them assist humans for the daily life.

Acknowledgements

Here, I would like to express my gratitude to all those who gave me the possibility to complete this thesis.

I am deeply indebted to my supervisor Dr. Taku Komura from the University of Edinburgh whose help, stimulating suggestions and encouragement help me throughout the research and writing of this thesis.

I have furthermore to thank Dr. Chiew-lan Tai from the Hong Kong University of Science and Technology for her valuable comments and suggestions in my research work. I would also like to thank Dr. Oscar Kin-Chung Au from the City University of Hong Kong for his insightful help, advice and consultation in my research work. Additionally I would like to thank Dr. Paul Turner from the Heriot-Watt Univeristy for his valuable advice in my research project.

Finally I would like to thank my former/current colleagues and fellow Ph.D students from the University of Edinburgh for their support in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially, I am obliged thank to Hubert Pak-Ho Shum, He Wang and Adam Barnett.

I also want to thank NAMCO BANDAI Games Inc. for providing the game motion data used in my research work.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Shu Lim Ho*)

To my beloved family

Table of Contents

List of Figures

Chapter 1

Introduction

Character motion synthesis is an important research topic which has a wide range of applications including films and computer games. One of the grand challenges is synthesizing motions that involve tangling and persisting contacts such as those observed in wrestling and dancing, and motions in constrained environments such as riding on a car.

Currently, such scenes are manually designed by highly experienced animators. A basic technique called keyframe animation is mostly used for such a purpose. The animators manually design the postures of the characters using forward and inverse kinematics and then the movements are produced by interpolating the joint angles of the characters. Careful editing is needed for animation of multiple characters in close proximity as they can easily result in visual artifacts such as penetrations and pass throughs of each body part. The animator needs to design many keyframes and the movements are difficult to be recycled for different conditions. As a result, this process is inefficient and costly.

Using motion capture systems (MOCAP) for automating this process is not an easy task. Optical systems, that make use of stereo vision technologies, suffer from occlusions of the markers due to the number of people in the environment. Inertial, magnetic or mechanical systems can be damaged / injure the subjects when strong impulses are exchanged between the subjects. Combining individually captured human motion data can be one of the solutions. However, again, this is a daunting task even for an experienced animator as each movement easily results in penetrations of body parts, as there is no coordination of the bodies when they are captured.

The main problem of existing techniques in motion editing and synthesis lies in the conventional representation of the character states that is based on the joint angles. Let

us call the manifold composed of valid motions by humans without any penetrations or pass throughs by the body as *motion space* and that defined by conventional parameters such as joint angles as *joint angle space* for the rest of the thesis. Two spaces are called isotopic when continuous functions that map elements of one space to the other exist. The motion space and joint angle space are not isotopic, because a continuous motion in the joint angle space does not necessarily become continuous in the motion space. Let us consider such an example. Given two keyframe postures shown in Figure 1.1, if we linearly interpolate them by joint angles, the arms of the characters will pass through each other as shown in the bottom of Figure 1.1. On the other hand, the blue arrows in Figure 1.1 indicate the desired transition motion. This example highlights

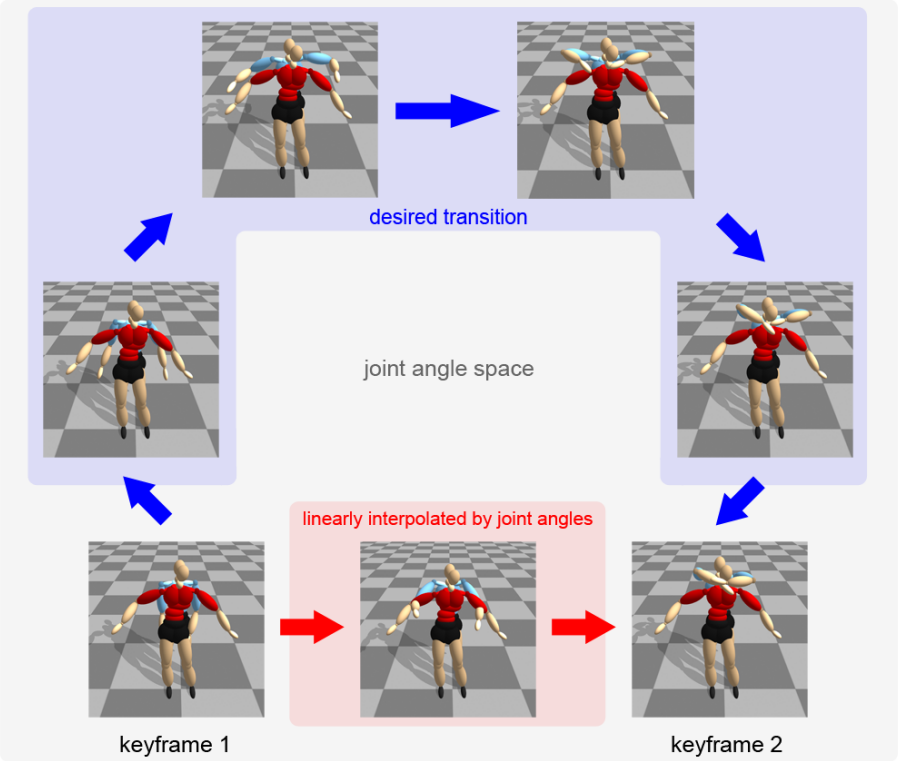


Figure 1.1: Linearly interpolating the keyframes by joint angles and the desired transition motion.

the problem when designing motions based on a representation that is not isotopic to the real world.

1.1 Research Area

In order to tackle these problems, we search for a new representation that is isotopic to the motion space, which means that a continuous movement defined in the new space can be mapped to a valid movement in the motion space and vice versa.

In fact, such a representation can be found in knot theory, where two knots are considered equivalent if one of them can be smoothly deformed into another without cutting the strings. Many concepts and techniques in knot theory can be applied for articulated characters that have tree structures. We develop a representation that is based on such techniques and apply them to the following areas in character animation.

1.1.1 Human Motion Indexing and Retrieval

A representation based on techniques in knot theory has features that are robust against kinematic difference, which is effective for motion indexing and retrieval of characters in close proximity. Human motion indexing and retrieval are important for animators due to the demand to search for motions in the database which can be blended and concatenated. Most of the previous researches on human motion indexing and retrieval compute the Euclidean distance of joint angles or joint positions. When selecting motions of close interactions, animators are interested in the way how the characters interact with each other rather than low-level parameters such as joint angles. In addition to that, joint angle metric suffers from the problem mentioned above - motions similar in the joint angle space can be actually very different in the motion space. Using our new representation, we can define a distance metric that is along the manifold of motion space.

We compute and encode how the two bodies are tangled with each other using Gauss Linking Integral (GLI) [Pohl, 1968] and rational tangles [Kauffman and Lambropoulou, 2003]. The encoded relationships, which we define as *TangleList*, are used to determine the similarity of pairs of postures. By using such an abstract representation, we can efficiently and effectively categorize semantically similar motions of multiple characters in close contacts even though the motions are quite different at the level of generalized coordinates (joint angles or joint positions) (Figure 3.1). This representation defines a new distance metric for calculating the similarities of multi-character interactions.

Using the new topology-based state representation, we can index and retrieve motions such as one person piggybacking another, one person assisting another in walk-

ing, and two persons dancing / wrestling. The new method is useful to manage a motion database of multiple characters. We can also produce Motion Graph structures of two characters closely interacting with each other by interpolating and concatenating movements, which are useful for 3D computer games and computer animation.

1.1.2 Motion Synthesis

Having presented a new representation and distance metric in motion space, next we present a method to synthesize a motion along the manifold in the motion space. There are many traditional techniques for synthesizing character motions. For example, Inverse Kinematics (IK) has been widely used in editing motions of the characters and robots in computer animation and robotics communities. However, most of the existing methods directly edit the joint angles or joint positions without considering the structure of the motion space. As a result, motions with interpenetrations of body segments will be produced when the characters are closely interacting with each other.

Here, we introduce a new coordinate system called *Topology Coordinates*, based on the representation of tangles that we used for motion indexing and retrieval. Using Topology Coordinates, we can directly edit and plan close interactions in motion space. This allows us to compute the motion by using local search methods instead of global path-planning algorithms. As a result, the computation for collision avoidance can be greatly reduced for complex motions such as tangling the segments of the body.

Topology Coordinates can be used with prevalent techniques in computer animation, such as keyframe animation and inverse kinematics. We show that character motions that involve close contacts such as wearing a T-shirt, passing the arms through the strings of a knapsack, or piggyback carrying an injured person can be synthesized efficiently.

1.1.3 Motion Adaptation and Deformation Transfer

Next we propose a new representation for handling motions without many tangles and enhancing our idea of topology-based motion synthesis to 2D manifold surfaces such as meshed characters, cloth models and 3D environments composed of polygons. The new representation is called *Interaction Mesh*, which encodes the structure of the open space between the nearby body parts. The Interaction Mesh provides a unified treatment for interacting body parts of single or multiple characters as well as objects in the environment. As a result, it is applicable to many types of scenarios, such as

when a single character's actions involve close interactions between different body parts (dancing), multi-character interactions (wrestling, fighting games) and characters in constrained 3D environments (riding on a car). Additionally, the motions may either involve much tangling and contacts (e.g. judo, Figure 5.1) or little contact (e.g. Lambada dance). The idea of Interaction Mesh is general enough to be applied for 2D polygon structures, which extends its applicability for deformation transfer of 2D mesh surfaces.

1.2 Contributions

The contributions of this thesis can be summarized as follows:

- We propose a new topology-based representation for describing the relationship between two characters who are closely interacting with each other. The entanglement of two bodies are computed and encoded based on Gauss Linking Integral and the concept of rational tangles in 3D space.
- We propose the concept of *Topology Coordinates*, in which the spatial relationships of the segments are embedded into the attributes. Editing and planning motions of closely interacting characters by Topology Coordinates can greatly reduce the computation for collision avoidance using local search methods.
- We propose the concept of *Interaction Mesh*, in which the spatial relationships between closely interacting body parts of articulated characters and objects in the environment are encoded. We further propose an automatic method for editing / retargetting close interactions of characters in different morphologies using the Interaction Mesh.

1.2.1 Publications

Portions of the work presented in this thesis have previously been published in the following academic papers:

- Wrestle alone: Creating tangled motions of multiple avatars from individually captured motions - *Edmond S. L. Ho and Taku Komura*
  Proceedings of Pacific Graphics 2007, pp. 427-430, Oct 2007.
  Bibliography reference [Ho and Komura, 2007b]

- Planning tangling motions for humanoids - *Edmond S. L. Ho and Taku Komura*
  Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp. 507-512, Nov 2007.
  Bibliography reference [Ho and Komura, 2007a]

- Character Motion Synthesis by Topology Coordinates - *Edmond S. L. Ho and Taku Komura*
  Computer Graphics Forum (Proceedings of Eurographics 2009), vol. 28, issue 2, pp. 299-308, Mar 2009.
  Bibliography reference [Ho and Komura, 2009a]

- Indexing and Retrieving Motions of Characters in Close Contact - *Edmond S. L. Ho and Taku Komura*
  IEEE Transactions on Visualization and Computer Graphics, vol. 15, issue 3, pp. 481-492, May/June 2009.
  Bibliography reference [Ho and Komura, 2009b]

- Real-time Character Control for Wrestling Games - *Edmond S. L. Ho and Taku Komura*
  Springer Lecture Notes in Computer Science (Proceedings of Motion in Games 2009), LNCS 5884, pp. 128-137, Nov 2009.
  Bibliography reference [Ho and Komura, 2009c]

- Spatial Relationship Preserving Character Motion Adaption - *Edmond S.L. Ho, Taku Komura, Chiew-Lan Tai*
  ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010), vol. 29(4), July 2010.
  Bibliography reference [Ho et al., 2010]

- A Finite State Machine based on Topology Coordinates for Wrestling Games - *Edmond S. L. Ho and Taku Komura*
  To appear in Journal of Computer Animation and Virtual Worlds.
  Bibliography reference [Ho and Komura, 2010]

1.3 Thesis Structure

In Chapter 2, we first review related work in computer animation and robotics. Here we focus on methods to synthesize motions of multiple characters in close proximity.

We discuss how difficult it is to solve our problem by existing methods and why we need to introduce a topology-based representation. We also review techniques in knot theory that we use in this thesis. Finally, we review the previous work in mesh editing and deformation to explore representations for handling 2D manifolds.

In Chapter 3, we propose a new data structure called TangleList based on the concept of GLI and rational tangles in knot theory to encode the relationship of two characters in close contacts. The experimental results show that users can retrieve semantically similar motions/postures accurately by using the new topology-based distance metric. In addition, the results show that the new distance metric can be integrated into an existing data-driven motion editing method, such as the Motion Graph, and it can effectively avoid inter-penetration of the different body parts.

In Chapter 4, we introduce the concept of Topology Coordinates for motion synthesis. We also show an example of applying Topology Coordinates for real-time wrestling games. The experimental results show that the new method can efficiently produce motions that involve complex tangling. The new method also outperforms existing global path-planning algorithms for planning the motions of characters in close contacts.

In Chapter 5, we propose a new data structure called Interaction Mesh for encoding the spatial relationship between closely interacting body parts of articulated characters and objects in the environment. We will also present an automatic method for motion editing / retargeting which preserves the spatial relationships embedded in the Interaction Mesh.

In Chapter 6, we conclude the work in this thesis and discuss about the possible future directions.

1.4 Definition of Terms

Space-time constraint is a method to synthesize realistic animation by spatiotemporally optimizing the motion subject to constraints based on body positions, kinematics and dynamics. It solves the problem of high frequency movements which can appear when per-frame motion editing methods (e.g. Inverse Kinematics) are used.

Motion Graph is a method to synthesize smooth motion sequence by concatenating and blending captured motions. It is a large scale Finite State Machine (FSM) which the nodes correspond to postures and edges correspond to short motion clips. It can

be computed automatically from captured human motion data by computing the Euclidean distance of all the postures in the data and connecting them by edges if their distances are below a threshold. By traversing its nodes and edges, a new motion sequence will be created.

Laplacian Coordinates is a type of differential coordinates that represent the relative position of each vertex to its neighbouring vertices. Laplacian coordinates has been widely used in 3D geometry editing, as the relative representation can well-preserve the local details.

Writhe is the total number of crossings found when projecting the link(s) from 3D space onto the 2D plane.

A **Tangle** is composed of two or more disjointed strings which are twisted around each other and whose end-points are fixed in the 3D space.

2-tangle is a tangle composed of 2 strings.

Rational tangles are a group of tangles which can be composed of successive twists of two parallel strings around the vertical and horizontal axes. A rational tangle is also a 2-tangle.

Generalized coordinates are a set of parameters that can describe the configurations of a system. In this thesis, the generalized coordinates of the characters or humanoid robots are joint angles.

Topological relationship is the term used in this thesis to describe the abstract spatial relationship of two different objects, which is not affected by the coordinate system or the spatial geometric details. We especially refer to how body parts of articulated objects are tangled around each other.

Writhe matrix explains how much each pair of segments from two strands/links contributes to the total writhe value. Given two chains/links $S1$ and $S2$, each composed of m and n segments, we can compute a $m \times n$ matrix which contains the writhes of every pair of segments.

Meshes have been widely used for representing 3D objects in Computer Graphics. A mesh is composed of vertices, faces and edges. These information can be used for reconstruct the shape of the objects. Polygon mesh and volumetric mesh are common data structure in shape modelling and manipulation in Computer Graphics. A polygon mesh is used for representing the surface of a 3D object. A volumetric mesh does not only represent the surface but also the interior/internal structure of the object.

Chapter 2

Related Research

The aim of this research is to synthesize character movements that involve a lot of close interactions with other characters, objects or the environment. We also aim to interactively control the characters under such an environment. This is a tough problem as it can easily result in collisions and penetrations of the body parts.

Historically, such motions have been manually designed by animators with special skills using tools based on forward / inverse kinematics. Once the keyframes are designed, the movements in between are computed by linearly interpolating the joint angles of the body.

As manual motion synthesis is costly and labour intensive, methods to make use of captured motion data to simulate interactions among multiple characters have been developed. We first review such data-driven approaches in Section 2.1.

On the contrary, methods to automatically synthesize the movements based on physical simulation, optimization and path planning is another major stream of research in character animation. We will next review such automatic methods which are applicable to close character animation in Section 2.2. We will also review hybrid methods that combine the data driven approaches and automatic methods.

Both the data driven approach and the automatic synthesis approach have difficulties synthesizing motions that involve close proximity due to problems of collisions and penetrations. We conclude that the fundamental problem lies in the representation of the state that is based on joint angles. Further discussion can be found in Section 2.3.

For describing a scene, a representation based on the spatial relationship of different body parts is required. In fact, such a representation can be found in knot theory. We review the basics of knot theory and the techniques which can be applied to our research in Section 2.4.

As knot theory mainly focuses on the spatial relationship of 1D manifolds such as strands, there are problems extending the techniques for characters composed of 2D surfaces. There are also problems handling motions that does not involve complex tangling. Therefore, we explore representations that can handle 2D manifolds in Section 2.5. We mainly review techniques in mesh editing and deformation transfer, which are the two main topics of research in meshing today.

2.1 Data-driven approaches for simulating close character interaction

Here we review methods that we can make use of the captured motion data in order to simulate the close interactions of multiple characters. A simple approach for creating a scene of multiple avatars is to capture their motions altogether in the studio. We first review such researches in Section 2.1.1. As there are many difficulties for capturing the motions of multiple persons, another stream of data driven approach to simulate character interactions is to capture the motions of a single person and simulate the interactions by combining such motions. We review such approaches in Section 2.1.2. If we simply play back the captured motions, the movements are repetitive and monotonic. In order to cope with this problem, many methods to synthesize new motions by editing and interpolating existing data have been proposed. We review methods which are applicable to close character interactions in Section 2.1.3. When interpolating motions, one of the important issues is to automatically search for similar motions in the database and categorize them. We review such methods for character animation in Section 2.1.4.

2.1.1 Capturing movements of multiple persons

Here we review researches that take the straightforward approach to synthesize the animation of multiple persons by capturing their movements altogether at once. The motion data in the basketball game NBA LIVE 06 [Electronic Arts. Inc., 2006] are obtained in such a way.

Several researches in the area of crowd simulation also take such an approach for synthesizing the close interactions of pedestrians [Lee et al., 2007, Lerner et al., 2007, Paris et al., 2007]. Lee et al. [Lee et al., 2007] use an overhead camera to track the way pedestrians avoid each other in the crowd and propose to learn such behaviour using

non-linear regression. Lerner et al. [Lerner et al., 2007] assume collision avoidance can be represented by a restricted number of rules and learn them from the video data. Paris et al. [Paris et al., 2007] capture the trajectories of pedestrians under a number of specific situations such as passing through a restricted area like a door, walking in a narrow corridor and two streams of people crossing each other. The captured data are used to simulate the interactions of pedestrians in the synthesized scene. As only the 2D position and orientation of each person are tracked in these researches, these techniques are only applicable for simple interactions such as queuing and avoiding.

Some researches simulate very close interactions such as fighting in this stream. Park et al. [Park et al., 2004] capture the motions of two Taekwondo fighters and use Hidden Markov Models (HMM) to learn the transition between the states. Kwon et al. [Kwon et al., 2008a] capture the motion of two kick boxers and use probabilistic models to learn their behaviours.

Several problems occur when capturing the motions of multiple persons making very close interactions. First, it is difficult to capture realistic interactions due to the limitation of the capturing device and the psychological status of the subjects. The subjects need to wear intrusive devices which affect the psychological status and the physical performance of the subjects. There are also motions such as those in wrestling, which are difficult to be captured due the limitations of the tracking system - there will be a lot of occlusions when the optical system is used and the sensors might be damaged or broken if mechanical / inertial / magnetic systems are used. Also, the devices might endanger the subjects being captured. Second, the combination of actions that can be captured tend to be limited (due to the large number of possible combinations and physical limitations), and the combinations synthesized by their systems are therefore only a small subset of all possible combinations. Therefore, this approach is not very practical for interactions that involve close proximity and leads to research of combining individually captured motions.

2.1.2 Combining individually captured motions

Here we review methods to combine motions of singly captured motions in order to simulate the interactions of multiple characters. We first review the Motion Graph [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002], which is a basic approach to make use of captured motion data to interactively control characters. Next, we explain how we can simulate the interactions of multiple characters by controlling

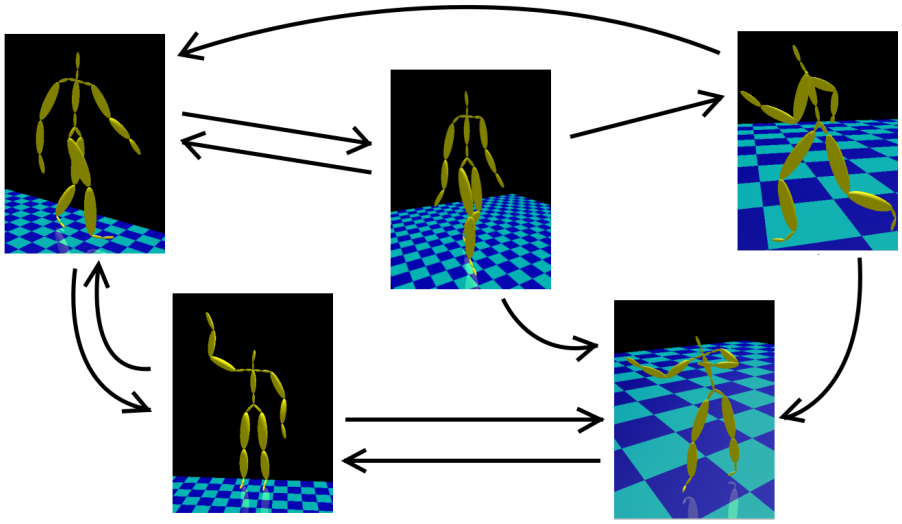


Figure 2.1: A Motion Graph.

each character using the Motion Graph.

2.1.2.1 Motion Graphs

One of the main objectives of researches in character animation is to achieve interactive control of the characters. This can be achieved by Motion Graphs [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002], which is a large scale Finite State Machine, whose nodes correspond to postures and edges correspond to short motion clips (see Figure 2.1). The Motion Graph can be automatically produced from captured human motion data by computing the Euclidean distances of all the postures in the data and connecting them by edges if their distances are below a threshold. Using the Motion Graph, a new series of movements can be synthesized by traversing along its nodes and edges. The user can interactively control the character by specifying what kind of motions he / she wants to see. Search trees can be expanded along the graph to find the best series of motion clips that satisfies the demand of the user.

Planning motions over the Motion Graph by expanding the search tree results in exponential complexity, and therefore various methods are developed to reduce the cost. Kovar et al. [Kovar et al., 2002] propose to use branch-and-bound algorithm to reduce the computation. Shum et al [Shum et al., 2007] propose to use $\alpha - \beta$ pruning for a min-max search problem in a competitive environment. Lau and Kuffner [Lau and Kuffner, 2005] and Kwon and Shin [Kwon and Shin, 2005] propose to use a small scale Finite State Machine rather than a large scale Motion Graph.

Another choice to enable interactive control of the characters is to do precomputation. Lau and Kuffner [Lau and Kuffner, 2006] extend their method by precomputing the most efficient series of actions for reaching a large area in the front. Shum et al. [Shum et al., 2008a] propose a patch-based approach that precomputes the meaningful interactions by expanding the game tree in a short horizon according to the specifications provided by the animator and saves them as a data structure called interaction patches. Then, the interaction patches are spatiotemporally concatenated during runtime to synthesize a large scale scene where many characters interact with one another.

Various methods to control the characters intelligently under the framework of reinforcement learning [Sutton and Barto, 2005] have also been proposed. In reinforcement learning, the system explores the state space and computes a state-action table which records the action that benefits the character most in the future. During run-time, the character simply needs to lookup the state-action table and launch the recorded action. More specifically, at each time step i, suppose the avatar selects an action and gets a reward defined by r_i. The optimal policy offers an action at every state that maximizes the following return value:

$$V = \sum_{i}^{N} \gamma^i r_i \tag{2.1}$$

where $0 \leq \gamma \leq 1$ is a constant called discount factor. Lee et al. [Lee and Lee, 2004] simulates a scene of two boxers fighting with each other by using temporal difference learning, which is a basic technique of reinforcement learning. Graepel et al. [Graepel et al., 2004] let the system learn how the characters in fighting games should move to compete well with user players.

Although reinforcement learning is an excellent approach to make the characters interact wisely with the other characters, there are many fundamental problems that must be solved when applying it for multi-character control. Basically, the dimensionality of the state space cannot be very large as the computational cost increases exponentially with respect to the dimensionality. This makes it difficult to handle multiple characters as the dimensionality becomes very large. Shum et al. [Shum et al., 2008b] reduces the amount of exploration by assuming the important area concentrates in where the two characters are facing each, and simulate two characters fighting and carrying luggage. Treuille et al. [Treuille et al., 2007] represent the value function as a linear sum of low-resolution basis functions and optimize their coefficients to make the pedestrian characters avoid the others. Different types of interactions require manual

tuning for reducing the state space, which hinders the generality of applying reinforcement learning for arbitrary interactions. Second, we come back to the problem of state representation discussed in Chapter 1. If we use a representation that is based on joint angles at the lowest level, we will result in a lot of penetrations and collisions. In summary, if we want to apply reinforcement learning to control the characters wisely, we must come up with a compact representation that can avoid the large state space problem and does not suffer from the huge computation of collision detection. Indeed, this is one of the objectives of our research.

2.1.3 Editing Captured Motions

When combining the individual motions to simulate the close interactions, each motion needs to be carefully edited so that the interaction looks realistic. For example, when synthesizing a scene that one character tries to hit another character who dodges the attack, the hand of the attacker must move towards the head of the defender and the defender must dodge it at the correct timing. As the number of captured movements are limited, it is necessary to edit the available motions to simulate realistic interactions between the characters. In this subsection, we review two basic methods to adapt the captured movements which are applicable to multiple character animation.

2.1.3.1 Editing Motions by Inverse Kinematics and Spacetime Constraints

The most basic approach is to apply inverse kinematics and edit the motion per frame. Given the new target locations / trajectories of the body parts, the captured motion can be edited to follow them. In fact, Lee and Lee [Lee and Lee, 2004] and Shum et al. [Shum et al., 2007, Shum et al., 2008b] apply such an approach to edit the trajectories of the interacting body parts. Gleicher [Gleicher, 1997] applies spacetime optimization to spatiotemporally optimize the motion to follow the target trajectories. It also solves the problem of high frequency movements which can appear when per-frame methods are used. The approach has also been applied for motion retargeting [Gleicher, 1998], which is to use the captured motions for characters of different body sizes. Most of the motion editing approaches do not consider the collisions of the body parts, which is essential when handling close character interactions. Some researchers propose to apply collision detection and push out the overlapping parts of the bodies [Lyard and Magnenat-Thalmann, 2008] in the direction of the normal vectors of the colliding surfaces. However, although collision detection can avoid interpene-

tration of body segments, the movement of one segment does not affect the movements of the other segments until collision occurs. Moreover, the normal vector may repulse the body in the direction of a dead end. As a result, the optimization process can also get stuck in a local minimum where the resulting motion is not penetration free.

2.1.3.2 Motion Interpolation and Parameterization

Another approach to adjust the motion is to interpolate two different motions to synthesize a new motion in between. This is known as an old technique called motion blending. Due to the large sample of motions available nowadays, many techniques to automatically search for similar motions and interpolate them have been developed. Rose et al. [Rose et al., 1998] propose a concept of *verbs* and *adverb* to generate new motion from examples. In their work, *verbs* refer to parameterized motions constructed from sets of similar motions, and *adverbs* are parameters that control the *verbs*. For each *verb*, the sample motions are time-aligned by manually specifying the key-time for every motions. Then, the motions clips are placed on a parameter space based on the characteristics of the motion clips. Motion blending is done by computing the weights of the sample motions in the corresponding *verb* using radial basis functions (RBF). By specifying the *adverbs*, new motion will be created. In addition, users can create a *verbgraph* so that transition motions between *verbs* can be generated. Figure 2.2 shows the sample motions (green) and the blended motions (yellow) created by their method. Park et al. [Park et al., 2002] propose a method to generate locomotions based on motion blending technique in real-time. New locomotions can be generated by specifying the parameters: *speeds*, *turningangles*, and *styles*. Like [Rose et al., 1998], RBF is used to compute the weights of the sample motions to blend new motions. Figure 2.3 shows the locomotion generated by their method.

One of the limitations of this method is that the similar samples are grouped manually. When the motion database is getting larger, it is time-consuming to classify the sample motions. Kovar et al. [Kovar and Gleicher, 2003] propose a method to automatically extract the relationships involving in timing, local coordinate frame and constraints of the input motions. They extend and apply the work to parametric motion blending in [Kovar and Gleicher, 2004]. As a result, logically similar motions can be blended to create a new motion. No manual work is required in the whole process. Mukai and Kuriyama [Mukai and Kuriyama, 2005] considers the problem of motion interpolation as a statistical prediction of missing data and remove the artifacts which

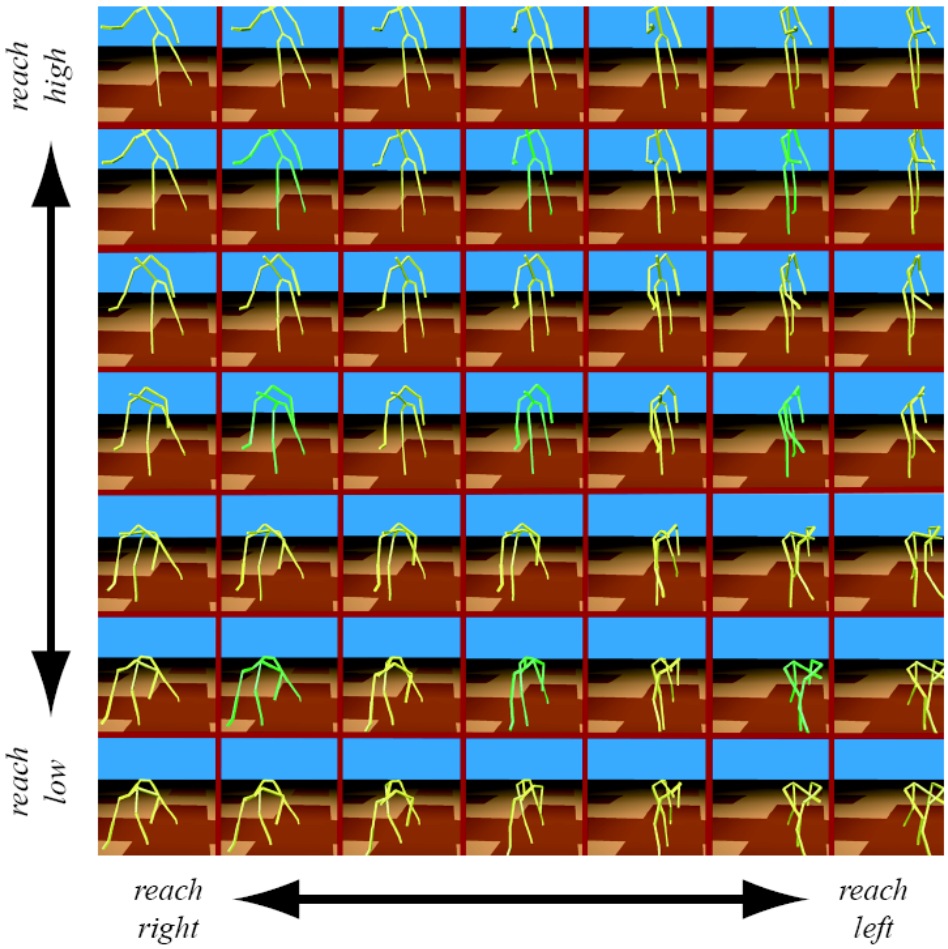


Figure 2.2: Results obtained in [Rose et al., 1998], the sample motions (green) and the blended motions (yellow). Reproduced from [Rose et al., 1998].

are present in [Kovar and Gleicher, 2004].

Approaches to interpolate similar motions have been extended to Motion Graphs. Shin et al. [Shin and Oh, 2006] and Heck et al. [Heck and Gleicher, 2007] propose to synthesize new motions by blending similar motions in the Motion Graph which start and end at common nodes. Since the number of nodes and edges in the graph is reduced, motion synthesis can be done quicker than the conventional Motion Graphs. The main difference between [Shin and Oh, 2006] and [Heck and Gleicher, 2007] is that [Shin and Oh, 2006] groups motion into the same group if they have a similar starting and ending poses (called *baseposes*) while [Heck and Gleicher, 2007] puts logically similar motions in the same node. The construction of Fat Graph [Shin and Oh, 2006] is illustrated in Figure 2.4. Safonova and Hodgins [Safonova and Hodgins, 2007] further optimizes the weights of blending to synthesize an optimal series of

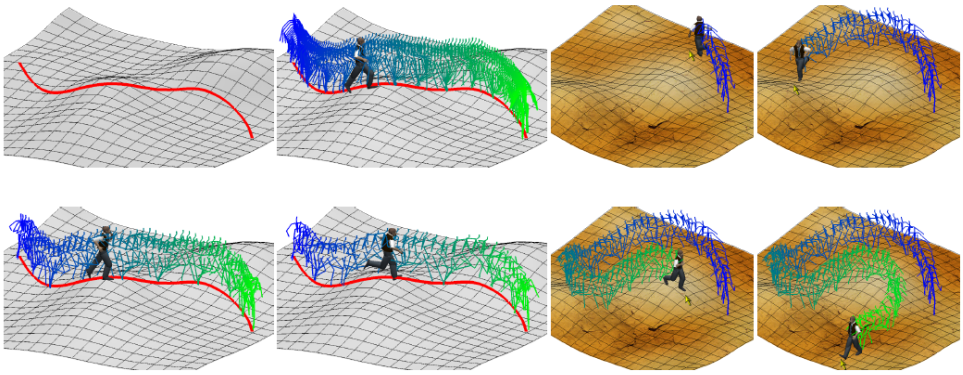


Figure 2.3: Locomotion on a terrain. Reproduced from [Park et al., 2002].

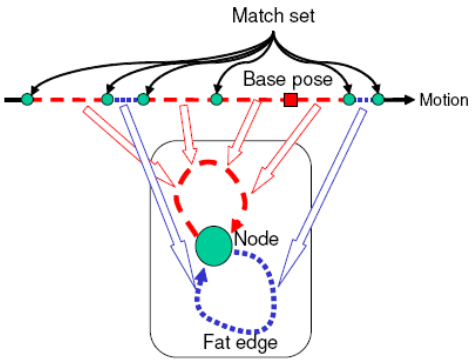


Figure 2.4: The construction of Fat Graph. Reproduced from [Shin and Oh, 2006].

movements according to the user input.

The main drawback of linear blending methods is that a large number of samples are required to produce arbitrary movements. Non-linear motion blending methods have been proposed to synthesize realistic motions from a limited number of samples. Grochow et al. [Grochow et al., 2004] propose to use Gaussian Process Latent Variable Model (GPLVM) [Lawrence, 2004] for interpolating postures and apply it for a non-linear inverse kinematics systems. Ye and Liu [Ye and Liu, 2010] propose to synthesize response motions when the character is pressed from different directions by using Gaussian Process Dynamical Models (GPDM) [Wang et al., 2008]. These techniques can reduce the number of samples for interpolation. The drawback is that the dataset must be selected carefully as they suffer from oversampling.

All parametric approaches require either the system or the user to classify similar motions to be blended. There are a number of methodologies proposed for such a purpose. We review such methods in the following section.

2.1.4 Content-based retrieval of human motions

Content-based retrieval of human motions is used to search for similar motions that can be blended together to synthesize a new motion. Arikan et al. [Arikan et al., 2003] propose using an Support Vector Machines (SVM) to classify the movements of characters. Feng et al. [Liu et al., 2003] propose building a hierarchical tree structure to quickly retrieve similar motions from the database. In order to consider temporal variations, Dynamic Time Warping (DTW) is used to synchronize the timings of actions [Kovar and Gleicher, 2003, Kovar and Gleicher, 2004]. Keogh et al. [Keogh et al., 2004] present scaling which is more efficient than DTW, and is more suitable for indexing large human motion databases. We categorize these methods as numerical methods, as the motions are classified based on distance of low level attributes such as the joint angles or 3D positions.

Most of the previous research in content-based retrieval of human motions are based on the Euclidian distances of the joint angles [Liu et al., 2003, Kovar and Gleicher, 2003, Kovar and Gleicher, 2004] or the 3D location of the joints [Arikan and Forsyth, 2002, Arikan et al., 2003]. Such metrics cannot distinguish motions of different context, which can easily result in interpenetrations of the body parts when they are linearly interpolated.

When handling close interactions, we are more interested in the semantic similarities rather than the numerical closeness of low level attributes. Müller et al [Müller et al., 2005, Müller and Röder, 2006] propose a semantic approach based on the correlation of four joint positions: a virtual plane is defined by the first three joint positions, and whether the last joint is in front or back of the plane, is used as an operator to index the posture. They select a number of combinations of joints which are effective to distinguish human motions and use them to index and retrieve human motions. Such an approach is more robust in retrieving semantically similar motions, as the results are not affected by deviation of low level attributes. They require the user to manually specify the feature that is suitable for each category of motion. This is not general enough for the retrieval of arbitrary motion. They also do not provide any methodology of interpolating semantically similar motions.

One of the objectives of this research is to propose a new representation that can easily distinguish semantically similar and different motions and also enable interpolations of the movements without suffering from collisions and penetrations of the body parts.

2.2 Automatic Character Motion Synthesis

Another direction of synthesizing close character interactions of multiple characters is to produce them automatically by applying physical simulation, optimization and path planning. Especially, motions that are dominated by physics, such as passive motions when falling down, are suitable to be produced by physical simulation. However, in most cases, we also need to control the characters that actively interact with the other characters. Such a control can be done by using PD control, or methods based on optimization. We first review such automatic methods based on physical simulation and optimization to synthesize motions of close character interactions in Section 2.2.1. When the motion involves lots of close contacts and avoidances, motion planning will be required to arrive to the target posture. We will review such methods in Section 2.2.2.

2.2.1 Character interactions by Physically-based simulation

Physically-based animation has been intensively applied for synthesizing character movements. Many computationally efficient methods of forward dynamics which has complexity of $O(n)$ (where n is the number of articulated bodies) have been developed [Baraff, 1996, Stewart and Trinkle, 1996], and are used in computer games through middleware such as Open Dynamics Engine [Smith, 2007]. They are most effective for passive movements such as characters falling down, which are often referred to as "ragdoll physics".

Now, let us think of synthesizing the interactions of two characters by using physically based simulation. Fighting scenes can be a good example. Movements of falling down when the character is hit can be well simulated by ragdoll physics. When we want the punched character to resist and recover the original posture, we can apply PD control [Zordan and Hodgins, 2002]. PD control is a method to apply torques to the joints relative to the discrepancy from the target posture:

$$\tau = k_p(\theta - \theta_d) + k_d(\dot{\theta} - \dot{\theta_d}) \tag{2.2}$$

where τ is the torque applied to the joints, θ are the generalized coordinates in the current frame, θ_d are their target values, and k_p and k_d are constants of elasticity and viscosity. We can further make the characters step out to keep balance by using hybrid methods that blend motions synthesized by PD control and captured motions of stepping out [Zordan et al., 2005, Arikan et al., 2005, Komura et al., 2005].

PD control is also effective to simulate active voluntary movements [Raibert and Hodgins, 1991, Wooten and Hodgins, 1996, Hodgins and Pollard, 1997, Yin et al., 2007], such as walkng [Yin et al., 2007], running [Hodgins et al., 1995] and playing sports [Hodgins et al., 1995, Wooten and Hodgins, 1996]. As tuning the gain parameters such as k_p and k_d is not very easy, van de Panne and Lamouret [Van De Panne and Lamouret, 1995] propose to first add external force to guide the character to the correct motion, and gradually reduce the guidance while optimizing the gain parameters such that they achieve a balanced gait motion. Yin et al. [Yin et al., 2008] propose to adapt the gain parameters for a normal walking motion to those under different conditions such as stepping over a large obstacle or pushing heavy objects, by applying a sampling-based optimization method. Wang et al. [Wang et al., 2009] optimize the gain parameters so that the motion has features similar to natural human gait motion. In such cases, Finite State Machines that are composed of a series of target postures are used to guide the characters to accomplish the desired motion. However, using such an approach for controlling characters when interacting is difficult because the PD controller can be very unstable when the characters unintentionally collide with others. In order to simulate motions such as wrestling or fighting that involve strong perturbation and impulses, it is necessary to adaptively change the stiffness of the joints through constant parameters of the PD controller according to the current status of the characters who are involved in the interaction.

Another stream of physically-based character animation is to parameterize the motions by generalized coordinates and spatiotemporally optimize an objective function based on physical parameters such as momentum [Witkin and Kass, 1988, Liu and Popović, 2002], acceleration [Popović and Witkin, 1999, Fang and Pollard, 2003] and joint torques [Cohen, 1992, Liu et al., 1994, Liu et al., 2005]. This is the same as the spacetime optimization method explained in Section 2.1.3. There the motions are adapted such that the kinematic constraints are satisfied. On the contrary, here the motions are edited such that dynamic constraints are satisfied. Liu et al. [Liu et al., 2006] simulate realistic interactions of two characters such as one character avoiding another and a mother pulling the hand of the child by iteratively applying such spacetime optimization to each character. The advantages of the spacetime optimization are (1) efficient motions that require minimal energy over time can be produced and (2) movements that prepare for future events can be produced.

Some researchers propose hybrid methods that make use of captured motion data and physical simulation. For example, Tak et al. [Tak et al., 2000] propose to con-

vert unbalanced captured motions into balanced motions by minimizing the integral of distance between the zero moment point (ZMP) and the supporting area during a gait cycle. If this distance is below zero throughout the gait cycle, the motion is balanced. Nishiwaki et al. [Nishiwaki et al., 2001] propose a similar method to synthesize a balanced motion of a robot. Some researchers propose per-frame methods. In Dynamics Filter [Yamane and Nakamura, 2003], the character is controlled so that its posture becomes similar to the reference captured motion while the body is balanced at every time step. Kagami et al. [Kagami et al., 2000] also balance the robots per-frame by such an approach. Shin et al. [Shin et al., 2003] propose such a method for character animation.

There are also hybrid methods for synthesizing response motions when hit or pushed. Zordan et al. [Zordan et al., 2005] propose to simulate the motion immediately after the impulse by ragdoll physics and then blend it to the most similar falling down motion saved in the database. Komura et al. [Komura et al., 2005] apply a similar approach but use a frame-based optimization approach that controls the linear / angular momentum of the body. Arikan et al. [Arikan et al., 2005] use the SVM to select the most appropriate motion data to be blended.

Although much research has been done for physically-based animation, they still suffer from the disadvantages due to the parameterization of the movements based on the joint angles, which we have discussed in Chapter 1. As a result, they still suffer from penetrations, pass-throughs and local minima that prevent them to be applied for close interactions. In order to synthesize a motion that is free of such artifacts, it is necessary to apply methods of global path planning, which is explained in the following subsection.

2.2.2 Global Path Planning

In order to automatically interpolate two arbitrary postures without falling into local minima, we need to apply global path planning methods. Methods such as Rapidly-expanding Random Trees [Lavalle and Kuffner, 2000] or Probabilistic Roadmaps [Kavraki et al., 1994] are applicable for such purposes. Both methods sample random postures in the state space and make connections between the samples if transitions can be made. Once the start and the goal are connected, path-planning is applied to search for the shortest path that connects the points. Although they efficiently sample points in the state space, the complexities are still exponential with respect to the degrees of

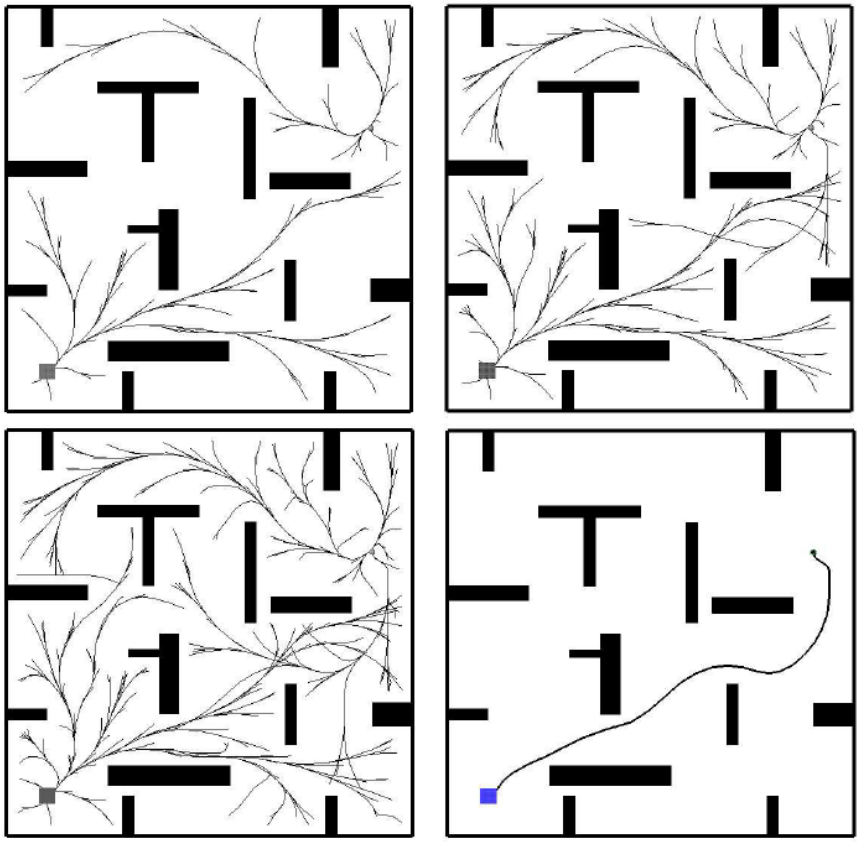


Figure 2.5: An example of expanding RRT in different stages. Reproduced from LaValle et al. [LaValle and Kuffner, 2001].

freedom (DOF). As a result, it is difficult to be applied to character animation unless the DOF is reduced. Yamane et al. [Yamane et al., 2004] compute the trajectory of the luggage to be carried by RRT and compute the character posture by inverse kinematics. This approach is not very general unless there is enough space for the character to freely move around. Shapiro et al. [Shapiro et al., 2007] use RRT to compute only the movements of the arm. This approach is only applicable for motions such as grabbing. Synthesizing interactions of multiple characters by global path planning is not easy as the DOF will double. Due to these problems, global path planning is rarely used for character animation purposes.

2.3 Summary of Character Animation and the Fundamental Problem to be Solved

We have reviewed two large streams of character animation ; the data-driven approach and the automatic motion synthesis approach. Although various movements can be simulated by these approaches, we will face problems when synthesizing close interactions of multiple characters. Among such problems, here we point out a few critical points that we cope with in this research.

1. Existing metrics (i.e., Euclidean distance of joint angles) to compute similarity of postures do not consider the spatial relationship between different body parts. As a result, motions of different context can be classified into the same group of motions, which results in penetrations / pass throughs when they are interpolated.

2. Existing motion synthesis methods plan the motions at the level of joint angles, whose complexity is exponential with respect to the DOF of the character(s). In addition to that, the evaluation of every transition requires intense computation of collision detection, which can slow down the process significantly.

We can summarize that the fundamental problem lies in the representation that is based on the joint angles. The joint angle representation does not include any information of spatial relationships between different body parts unless they are computed by forward kinematics. Spatial relationships are important for recognizing the similarity of motions and avoiding collisions. Therefore, we explore a new way to represent the human postures that considers the spatial relationship of different body parts. As we want to handle motions such as wrestling, which involve a lot of tangling, we first explore a representation in the knot theory in Section 2.4. Next, we explore methods to represent the relationship of two different 2D manifolds in the area of mesh editing and deformation transfer in Section 2.5.

2.4 Applying Knot Theory for Character Animation

Knot theory provides invariants which are useful for distinguishing one knot from another, which are useful for describing the way characters are tangled with one another. We first review the basics of knot theory in Section 2.4.1. In fact, such invariants have been applied for controlling robots to tie knots. We also review such research in Sec-

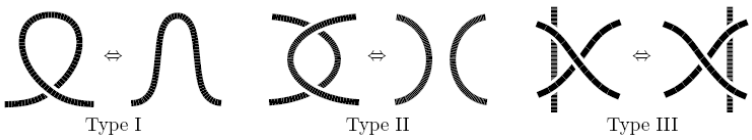


Figure 2.6: 3 types of Reidemeister moves.

Figure 2.7: Right-handed (a) and left-handed (b) crossings.

tion 2.4.1. Finally, we discuss theories and techniques that can be applied for character motion synthesis doing close interactions.

2.4.1 Basics of Knot Theory

Knots, which are closed, single strands, are considered equivalent if one of them can be smoothly deformed into another without cutting the strings. The deformation can be done by the Reidemeister moves [Reidemeister, 1935] as shown in Figure 2.6. As we need to handle open chains instead of closed loops, we use a concept called tangle [Conway, 1970] in knot theory. Among the tangles, there is a concept called 2-tangles [Conway, 1970], which is defined as a pair of strings whose end points are fixed in Euclidean 3D space.

Many invariants such as crossing numbers and polynomials [Adams, 1994] have been proposed to distinguish knots. However, none of them is proven to be always successful. The minimum crossing number is one of such invariants which is the sum of the crossings of the strands. Every crossing will be denoted as 1 or -1 as shown in Figure 2.7, depending on whether it is a left-handed or right-handed crossing. State spaces based on such invariants have been used to control robots to tie knots [Takamatsu et al., 2006, Wakamatsu et al., 2006, Matsuno et al., 2006, Saha and Isto, 2006, Saha and Isto, 2007]. In these researches, a state machine based on the crossings are prepared and used to guide the robots to manipulate the strands.

Most of the existing work based on knot theory assume that the knots and tangles

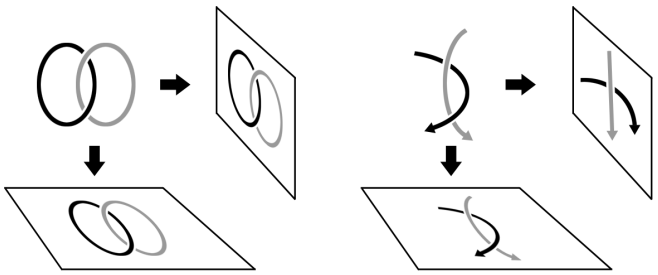


Figure 2.8: The projection plane does not affect the minimum crossing numbers for knots and links (left) but does for tangles (right)

are projected on a 2D plane before being analyzed. As knots or links are closed curves, the projection plane does not affect invariants such as the minimum crossing numbers (Figure 2.8, left). However, in case of tangles, the projection plane affects the number of crossings (Figure 2.8, right) and as a result, we cannot use the crossing numbers as the features. It is also difficult to define a plane to project the tangles onto, as the bodies are always moving around and their orientations will change from time to time. Therefore, a method to compute the tangles directly from the 3D location/orientation of the bodies is required in our research.

2.4.2 Gauss Linking Integral

Gauss Linking Integral (GLI), a discovery by Carl Friedrich Gauss in 1833, can directly compute the linking numbers of two separate closed curves from their 3D trajectories. GLI has been widely used in various research such as computing the topological similarity between different protein and DNA structures [Agarwal et al., 2002, Erdmann, 2004, Klenin and Langowski, 2000]. Since GLI can compute the writhing number in 3D, the topological structures of different protein and DNA can be analyzed in 3D. The GLI can also be applied for open curves such as 2-tangles. In that case, the output becomes the average number of crossings when viewing the 2-tangle from all directions [Erdmann, 2004].

In this research, we also extract the relationships between characters in 3D by computing the GLI between body segments, and guide the characters to tangle with each other. To the best of our knowledge, no similar research has been done in the area of computer graphics nor robotics.

The GLI of two directed curves γ_1 and γ_2 can be computed by

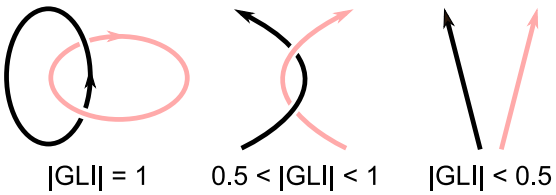


Figure 2.9: The GLI of two directed curves when one strand is surrounding the other (left), singly tangled (middle), and untangled (right)

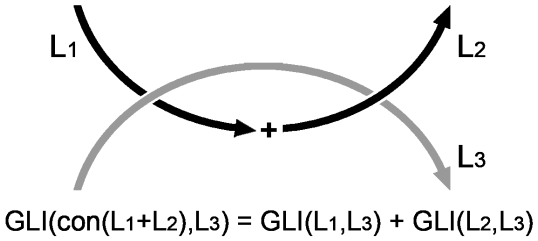


Figure 2.10: GLI satisfies the commutative rule

$$GLI(\gamma_1,\gamma_2) = \frac{1}{4\pi}\int_{\gamma_1}\int_{\gamma_2}\frac{d\gamma_1 \times d\gamma_2 \cdot (\gamma_1-\gamma_2)}{\|\gamma_1-\gamma_2\|^3} \tag{2.3}$$

where $\times$ and $\cdot$ are cross product and dot product operators, respectively [Pohl, 1968].

The GLI satisfies commutativity and is linear in summation; these are convenient features when computing all the tangles made between the segments:

$$GLI(L_1,L_3) = GLI(L_3,L_1) \tag{2.4}$$

$$GLI(L_1+L_2,L_3) = GLI(L_1,L_3)+GLI(L_2,L_3) \tag{2.5}$$

where L_1 and L_2 are two connected curves making a tangle with another curve L_3 (see Figure 2.10).

When calculating the GLI of long curves such as supercoiled DNA, the double integral in Equation (2.3) is computational costly. When the curve can be approximated by polylines, we can make use of the analytical solution proposed by Levitt [Levitt, 1983]. Given two polylines, $S1$ and $S2$, and they are composed of m and n line segments respectively, the total writhe (or GLI) of $S1$ and $S2$ can be calculated by

$$\sum_{i=1}^{m}\sum_{j=1}^{n} T_{i,j}$$

where $T_{i,j}$ is the writhe of segment i and segment j. Now we explain how to calculate $T_{i,j}$. Suppose points a,b and c,d are the end points of segment i and segment j, respectively. Let us define the vectors connecting a-b,a-c,a-d,b-c,b-d,c-d by $r_{ab}, r_{ac}, r_{ad}, r_{bc}, r_{bd}, r_{cd}$, respectively (Figure 2.11). Using these vectors, the normal

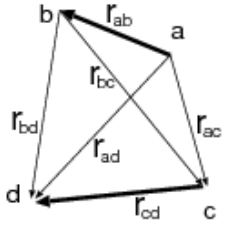


Figure 2.11: The tetrahedron composed by two line segments a-b and c-d.

vectors of the tetrahedron made by these four points can be calculated by:

$$n_a = \frac{r_{ac} \times r_{ad}}{|r_{ac} \times r_{ad}|}, n_b = \frac{r_{ad} \times r_{bd}}{|r_{ad} \times r_{bd}|}, n_c = \frac{r_{bd} \times r_{bc}}{|r_{bd} \times r_{bc}|}, n_d = \frac{r_{bc} \times r_{ac}}{|r_{bc} \times r_{ac}|}.$$

Finally, $T_{i,j}$ is calculated by

$$T_{i,j} = \arcsin(n_a n_b) + \arcsin(n_b n_c) + \arcsin(n_c n_d) + \arcsin(n_d n_a).$$

2.4.3 2-Tangles

As GLI only computes how many times the two curves are winding around each other, it does not provide the full information of the relationships between two chains. The concept of 2-tangles [Conway, 1970] can be applied for analyzing the topological relationships of the two open curves. 2-tangles can be categorized into prime tangles, self-knotted tangles, and rational tangles (See Figure 2.12). The rational tangles are a group of tangles which can be composed of successive twists of two parallel strings around the vertical (Figure 2.14 (a)) and horizontal (Figure 2.14 (b)) axes. Rational tangles can be used in our research because we can calculate an invariant called continued fractions to distinguish every tangle from the others. In addition, there are tangle operations for rational tangles as shown in Figure 2.13.

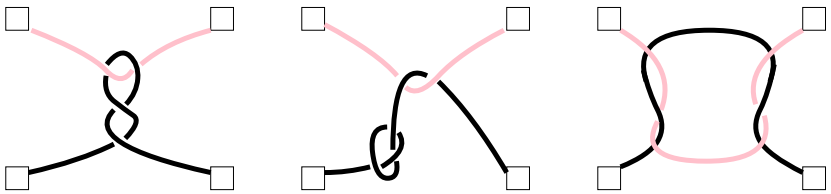


Figure 2.12: Examples of 2-tangles: (a) rational (b) self knotted and (c) prime tangles. The rational tangles can be composed by successive twists of 2 ends.

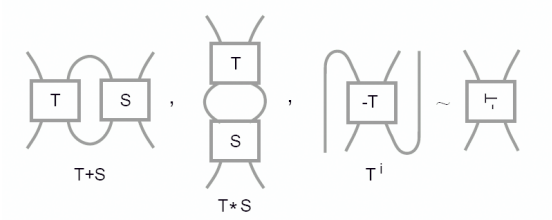


Figure 2.13: Addition, multiplication and inversion of 2-tangles. Reproduced from [Kauffman and Lambropoulou, 2004].

2.4.4 Conclusion and outlook

By extending the concept of rational tangles in tangle theory, it is possible to represent the topological relationship between the tangled human characters. The existing studies in knot theory assume that projected images with the minimum crossing number of the knots / tangles are available. However, by applying Reidemeister moves to a knot or tangle, tangle projections with different crossing number will be obtained although they are topologically equivalent. As a result, unless we have a systematic way to project the knot or tangle, it is difficult to extract the topological relationship between the tangled characters in 3D space.

Instead of analyzing the topological structure from a tangle projection, GLI can be used to detect if two curves are tangled in 3D space. In other words, computing the GLI between the body segments among the characters indicates whether they are tangled or not. By combining the extracted tangling information from the GLI calculation and the concept of rational tangles, we can distinguish the spatial relationship among multiple characters.

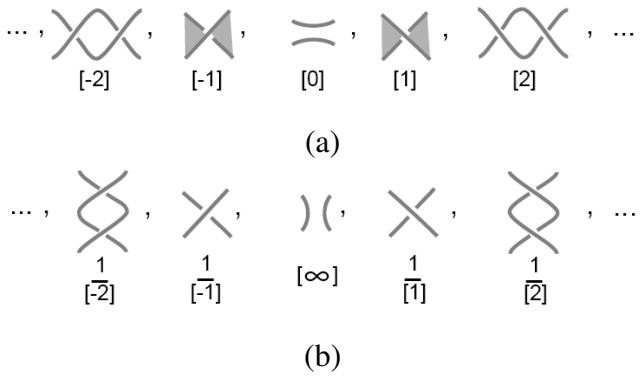


Figure 2.14: Examples of (a) integer tangles and (b) vertical tangles. Reproduced from [Kauffman and Lambropoulou, 2004].

2.5 A Representation for Multiple 2D Manifolds

The knot theory and GLI are adequate for handling motions involving tangles between 1D manifolds such as strands or skeletons. However, these techniques cannot handle close interactions without any tangles. Further, an extension to motions involving character shapes are difficult since relationships between rigid bodies or surfaces need to be encoded. Here we explore a representation for 2D surfaces such as mesh data. We will review two basic areas of geometric data processing ; differential coordinates-based mesh editing and deformation transfer of mesh data. Differential coordinates such as Laplacian coordinates enable users to edit 3D geometric shapes while preserving the local details of the original mesh structure. Deformation transfer [Sumner and Popovic, 2004] can produce an animation sequence of one character based on that of another. Although these areas have been extensively explored in the past, we argue that little research considers the spatial relationship of separate characters and objects. Such relationships are important for preserving the context of the scene while editing the character postures or transferring the original deformation to other characters.

2.5.1 Mesh Editing by Differential Coordinates

3D Mesh editing is an important area of research, which is needed to produce arbitrary mesh shapes from existing data. “As-rigid-as-possible” mesh editing [Alexa et al., 2000] decomposes 2D and 3D shapes into primitives such as triangles and tetrahedra, respectively, and tries to maintain the rigidity of the primitives while satisfying the constraints given by the user. However, preserving the rigidity of the shapes is computationally costly. Alexa [Alexa, 2003] propose to use Laplacian coordinates to

maintain the local mesh structure when editing the shape of the object. Sorkine et al. [Sorkine et al., 2004] extend this method by allowing rotation and scaling of the Laplacian coordinates. Differential coordinates are convenient for character animation as the animator can easily produce new postures of the deformable characters by simply dragging parts of the characters. One problem of the Laplacian techniques compared to "as-rigid-as-possible" techniques is that the shape of the character can be distorted due to volume loss. Zhou et al. [Zhou et al., 2005] produce a volumetric mesh by sampling new vertices inside the original mesh structure, and apply Laplacian mesh editing to the volumetric mesh. This method requires less computation and the volume loss problem can be avoided. When the character is not fully deformable, but is an articulated character with rigid body parts, it is necessary to add constraints that maintain the rigidity of the body segments. Adding such constraints into the process of solving the Laplacian deformation problem can slow down the convergence significantly. Shi et al. [Shi et al., 2007] solve this problem by using a cascading approach, that solves a sequence of optimization problems with different objective functions. Another problem when creating animation by Laplacian deformation is that the temporal coherence may not be preserved as the character shape is edited frame by frame. Xu et al. [Xu et al., 2007] propose a spacetime approach that preserves the temporal coherence by adding interframe constraints that minimize large movements every frame.

When applying these methods for animation of multiple characters in close proximity, the movement of one character will not affect those of the others until a collision between the body parts occur. This may be inconvenient in some cases. For example, when editing the movements of two characters dancing, the animator prefers both characters to move simultaneously while preserving the relationship between each other rather than each character changing its motion independently. However, no previous research in mesh editing focuses on this problem.

2.5.2 Deformation transfer

By using deformation transfer, an animator can make use of an existing sequence of mesh animation to control the movements of another character. Examples of such a scene is shown in Figure 2.15: In this example, the posture of a tiger is transferred to that of a cat.

The process of deformation transfer goes as follows: Using the basic rest-poses of the two meshes, the correspondence between the triangles are calculated. When a

Figure 2.15: A posture of a tiger transferred to a cat model. Reproduced from [Zayer et al., 2005].

new configuration of the source mesh is given, the rotation of the triangles from their original rest-poses are computed and added on top of the target mesh's rest-pose. This results in the rough transferred configuration of the target mesh, but the connectivity of the triangles are ignored at this moment. Finally, an optimization problem to maintain the connectivity of the triangles is solved to obtain the final configuration of the target mesh. For computing the correspondence, Zayer et al. [Zayer et al., 2005] use harmonic fields to compute the correspondence between the meshes.

One problem with these methods is that the spatial relationship of different body parts are not considered when editing the postures. Therefore, as you can see in Figure 2.15, the spatial relationship such as the limbs and the tail can be different after the deformation transfer. This can be a big problem especially in cases the morphology of the two characters are very different. For example, a motion to scratch the head done by a cat model can appear like scratching the neck bottom when it is transferred to a giraffe model. It is necessary to take into account the spatial relationship of body parts in order to maintain such context of the scene.

A recent work by Zhou et al. [Zhou et al., 2010] for deformation transfer [Sumner and Popovic, 2004, Zayer et al., 2005] represents the spatial relationships between multiple components of an object by Euclidean distances and encode them using a minimum spanning tree. Since the spatial relationships are assumed to be fixed (same as rest pose) during deformation, the method is not applicable to motions with time-varying spatial relationships.

2.5.3 Conclusion and outlook

Here we have explored new representations for multiple 2D manifolds. The representations such as differential coordinates only take into account the local details and does not preserve the relationship of separate body parts. The volumetric Laplacian representation by Zhou et al. [Zhou et al., 2005] is an interesting approach that tries to cope with the problem of volume-loss. However, it has not been used for preserving the spatial relationship of different objects. We will look into such an application in this thesis.

Deformation transfer represents the posture by the relative orientation of the triangles. Again, this does not consider the relationship of separate body parts and therefore can result in a change of the context after the posture is transferred to different surfaces. A very recent work by Zhou et al. [Zhou et al., 2010] is the first to consider the spatial relationships for deformation transfer. However, the application is very limited as they only consider animations in which the spatial relationship does not change much. In this thesis we explore a new representation that can encode the relationship of multiple characters in close proximity and can also handle 2D surfaces.

Chapter 3

Indexing and Retrieving Motions of Characters in Close Contact

In this chapter, we introduce a new topological representation to encode the spatial relationship between tangled body segments of multiple characters using Gauss Linking Integral (GLI) and rational tangles in Knot Theory. With the new topological representation, we can index and retrieve motions such as one person piggy-backing another, one person assisting another in walking, and two persons dancing / wrestling. The experimental results show that our method is useful to manage a motion database of multiple characters. We can also produce Motion Graph structures of two characters closely interacting with each other by interpolating and concatenating topologically similar postures and motion clips, which are applicable to 3D computer games and computer animation.

Portions of this chapter have previously been published as [Ho and Komura, 2009b].

3.1 Introduction

There is a high demand for indexing and retrieving motion data efficiently so that animators can easily search for the motion they want in the database. Several methods have been proposed to search human motion data in the database by giving an example query. Such methods are targeted for motions of single characters, and most of them evaluate the similarities of the motions by comparing low level attributes such as the joint angles or joint positions.

On the other hand, in computer animations and games, there are many scenes where multiple characters are densely interacting with each other. For example, in wrestling,

the arms and legs of each character are tangled with those of the others' in a complex way. To index such motions, we cannot simply apply the same methods as for single characters as the correlations between the characters will be ignored. Suppose one character is holding the neck of another character, as shown in Figure 3.1. If we want to search and blend motions to these postures, we need to take into account the fact that the right arm of the black character is tangled with the neck of the gray character. Only motions that keep such relationships can be blended to the characters' motions. As previous indexing methods of human motions do not take into account the topological relationships of body segments, they will not work well for dense interactions of multiple characters.

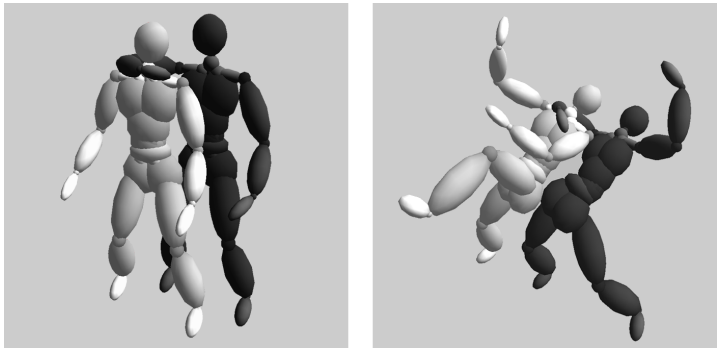


Figure 3.1: The topological relationship where the arm is tangled with the neck is the same for the above two postures, although the kinematic joint angles or 3D location of the joints are different

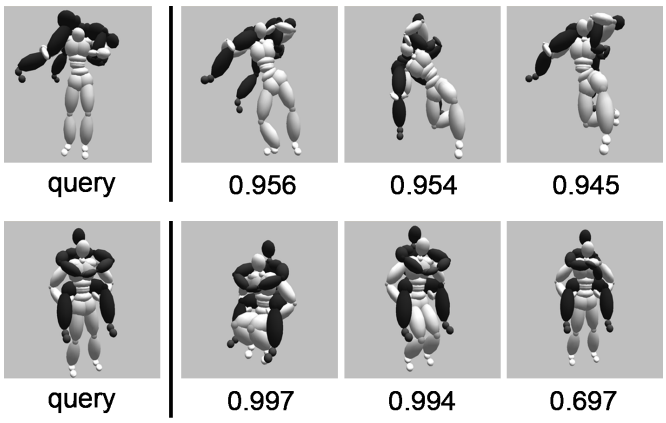


Figure 3.2: The human motion retrieval based on topological relationships: three pairs of postures similar to the query postures are returned. The values on the bottom are the normalized similarity of the output posture with the query posture.

In this research, we use tangles [Conway, 1970] made between the segments to

index the pair of postures of two characters. As there can be several tangles made by different segments of the body, we compose a data structure called a *TangleList* to represent the topological relationship of the two bodies. Given two TangleLists, we can compute their distance to evaluate the similarity of the set of postures. Then, it is possible to categorize the relationships of two characters, and give an example relationship as a query to search for similar pairs of postures, as shown in Figure 3.2. It is also possible to avoid interpolating / blending topologically dissimilar postures / motions which can cause body inter-penetrations. As a result, our method is also useful for applications such as motion synthesis.

This chapter is composed as follows: Section 3.2 explains how to extend the concept of tangles in knot theory to index, encode and compare the relationships between two characters. In Section 3.3, experiments are conducted to show the performance of using the topological relationship for content-based retrieval and human animation. In Section 3.4, we discuss the possibilities of applying the topological relationship and conclude the work in this chapter.

3.2 Representation and Comparison of Topological Relationships

In this section, the methodology to compute and encode the way two human bodies are tangled with each other is explained.

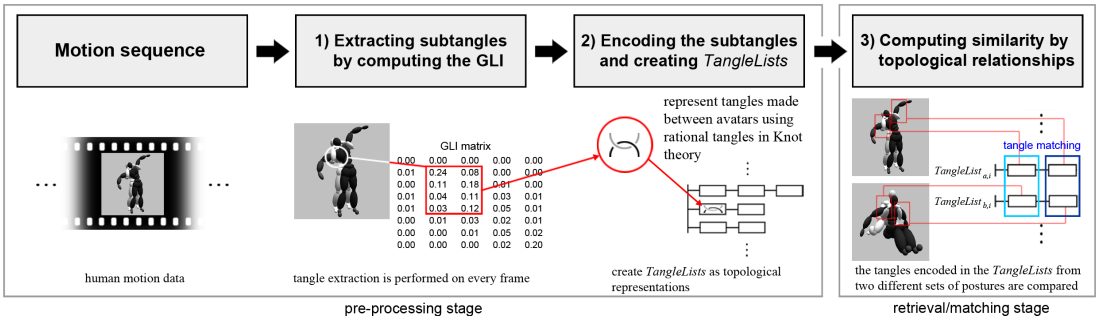


Figure 3.3: The process of encoding the tangled postures. For every path connecting the end effectors, we 1) compute and 2) encode the tangle information and 3) compare the results with those from other postures.

The overview of the methodology is shown in Figure 3.3. We first compute all the tangles made between the paths connecting the end effectors. The tangle information

is then encoded into a data structure called a *TangleList*. We calculate the similarities between two pairs of postures by matching the subtangles in the two TangleLists.

The rest of this section proceeds as follows: We first explain the concept of 2-tangles, which is the minimal unit for representing the topological relationship, and then explain how to represent the relationship of body structures by a set of 2-tangles. Next, we explain how to compute and encode the tangle information from the 3D postures of the characters. Finally, we explain how to compare two sets of postures based on the encoded data.

3.2.1 2-tangles

We use 2-tangles [Conway, 1970] as the minimal unit to represent the topological relationship of two characters. A 2-tangle is defined as a pair of strings whose end points are fixed in Euclidean 3D space.

Examples of 2-tangles are shown in Figure 2.12 (a)-(c). Most of the tangles of the bodies can be represented by 2-tangles, or a set of 2-tangles.

2-tangles can be categorized into rational tangles (2.12(a)), self-knotted tangles (2.12(b)), and prime tangles (2.12(c)). The rational tangles are a group of tangles which can be composed of successive twists of two parallel strings around the vertical and horizontal axes. In this research, we limit the tangles made between the paths to rational tangles, because (1) they are the most basic tangles which can represent most of the postures of humans tangled with each other, and (2) there is an invariant that can be used to distinguish every tangle from the others.

3.2.2 Tangles of tree structures

As we need to handle human characters, we have to compute the tangles made between tree structures. Trees are composed of edges and nodes, and therefore, tangles made between them will be more complex than those between single strings.

The tangles between trees can be examined by checking all the tangles made between the paths connecting the end effectors of the trees. The graph structure that is used to represent the human body in this research is shown in Figure 3.4. There are ten paths connecting the end effectors of this graph.

As shown in Figure 2.10, the GLI of two paths is computed by summing the GLI between every pair of segments on the paths. That means no matter how you divide the paths into many segments, the total GLI of the 2 paths will be the same by summing

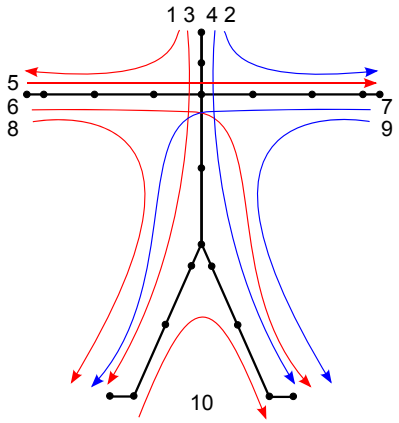
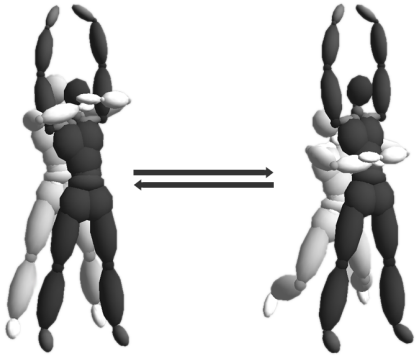


Figure 3.4: The tree structure of the graph that is used to represent the body structure. There are 10 paths that connect the end effectors.



(a) (b)

Figure 3.5: The homeomorphic Reidemeister moves after which the relationship must be considered equivalent.

up the GLI between every pair of segments. In order to speed up the calculation, we divide the paths at the joint positions to minimize the number of segments.

The topological relationships of two tree structures are considered equivalent when the 2-tangles of all the paths are equivalent. An example of a set of postures where the two characters are tangled is shown in Figure 3.5 (a). In this case, path 5 of the gray character is tangled with paths 3,4,6,7,8,9 of the black character. The advantage of representing the tangles of tree structures by the combination of all the 2-tangles is that, although the tangle crosses a node while the characters are moving, as shown in Figure 3.5(b), the relationship is acknowledged unchanged. This kind of translation of tangles is called Reidemeister moves in knot theory. It is important that the state of the tangle does not change under Reidemeister moves.

3.2.3 Detecting the Tangles

In this subsection, we explain how to find out the tangles directly from the 3D position / orientation of the body segments by computing the Gauss Linking Integrals.

In knot theory, it is always assumed that the tangles or knots are projected onto a 2D plane. As knots or links are closed curves, the projection plane does not affect invariants such as the minimum crossing numbers (Figure 2.8, left). However, in case of tangles, the projection plane affects the number of crossings (Figure 2.8, right) and as a result, we cannot use the crossing numbers as the features. It is also difficult to define a plane to project the tangles onto, as the bodies are always moving around and their orientations are changing from time to time. Here we explain a method to compute the tangles directly from the 3D location/orientation of the bodies.

3.2.3.1 Efficient Detection of Tangles by the GLI Matrix

This process is similar to what has been done in our previous work in [Ho and Komura, 2007b, Ho and Komura, 2007a]; however, instead of conducting double integrations of Gauss Integrals for every path as stated in Eq. 2.3, we do this more efficiently by representing the segments connecting the joints by line segments, and calculating the GLI between the line segments using the analytical solution [Levitt, 1983]. The readers are referred to Section 2.4.2 in the Related Work for the details. As a result, the GLI between arbitrary local paths can be simply calculated by summing the elements of the matrix.

Now we explain how to find all the tangles based on the GLI. Let us assume we are going to compute the rational tangles made between two serial links of rigid segments. This starts by computing all the subtangles made between the two links. Suppose the two links A and B are composed of m and n segments, respectively. An $m \times n$ matrix that contains the GLI between every pair of body segments is composed. Let us define this matrix as the GLI matrix. First, we find out all the minimal subtangles, for which the absolute sum is larger than 0.5 by scanning all the submatrices in the GLI matrix. The cost for scanning the GLI matrix is O(m^2n^2).

An example of a 2-tangle and its GLI matrix is shown in Figure 3.6. Five subtangles are found and the corresponding submatrices are surrounded by the rectangles.

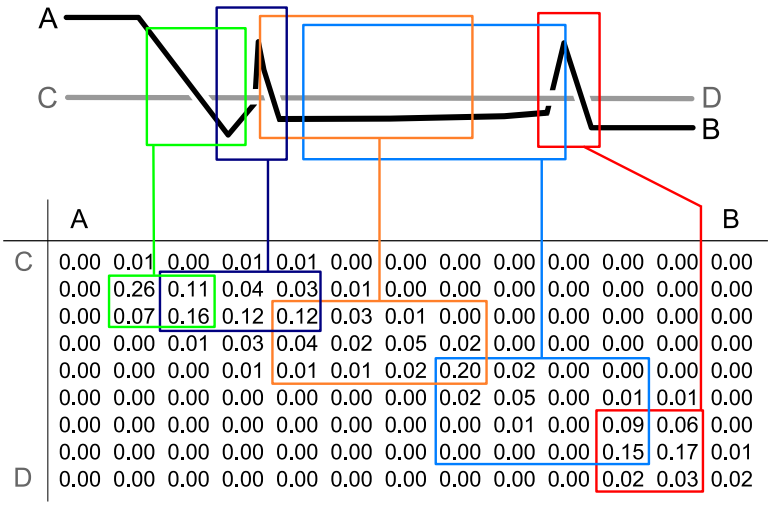


Figure 3.6: A 2-tangle and its GLI matrix. The five subtangles and their corresponding submatrices are visualized.

3.2.4 Encoding the Tangles

Let us define the rational tangle to be encoded by T. Rational tangles are composed of successive, integer numbers of horizontal twists (Figure 3.7 (a)) and vertical twists (Figure 3.7 (b)). The composition of the TangleList proceeds by untangling T, which

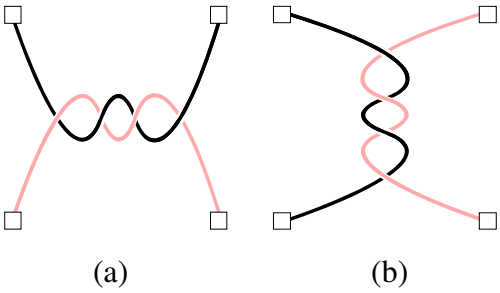


Figure 3.7: Examples of (a) horizontal tangles and (b) vertical tangles.

is done by successive twists of the two ends, as shown in Figure 3.8. In our case, we do not necessarily twist for an integer number of times, as the tangles are defined in the 3D space based on the location of the joint positions. Instead, we keep the GLI value of each twist. On the other hand, it is also difficult to define any absolute standards for vertical / horizontal twists as the bodies can be oriented in arbitrary ways. Therefore, we assume the initial twist is a horizontal twist.

In order to correctly untwist tangle T, we need to define the type of the subtangles. Let us assume a tangle T is composed of two strands, a and b, and the directions are defined in both strands. The subtangles S_i composing T can be categorized into 4

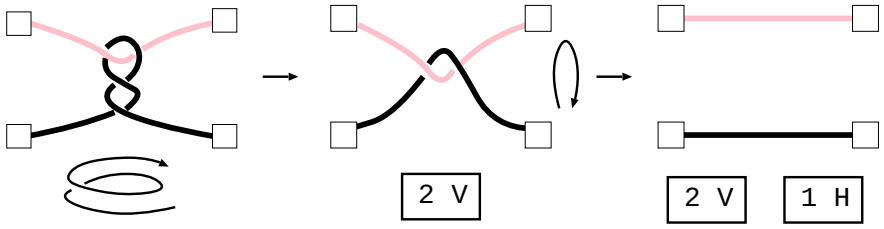


Figure 3.8: An example of encoding a rational tangle while untangling it. The tangle is encoded as "two vertical twists (2V) and one horizontal twist (1H)".

types: those composed by strand a and a (type AA), strand b and b (type BB), and by strand a and b in the forward direction (type AB), and in the opposite direction (type BA). The four types are shown in Figure 3.9.

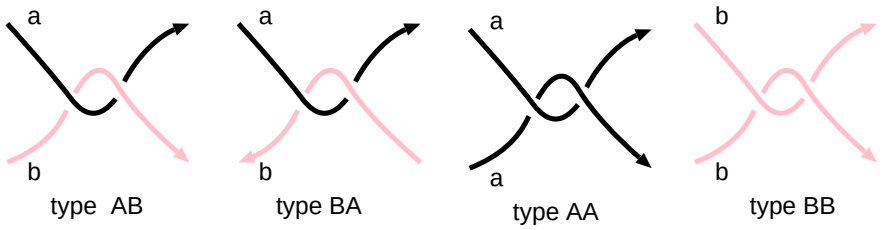


Figure 3.9: The four types of subtangles composing the tangle made by two strands a and b. We assume the directions of the strands are defined. Those composed of a and b (left most, left middle), only by a (right middle) and only by b (right most).

Now we can define three attributes for each subtangle, the *type*, *GLI*, and *twist*. The *type* is either of *BB*, *AB*, *BA*, *AA*, the *GLI* keeps the GLI value of the subtangle, and *twist* tells whether the subtangle is either made by a "vertical"(V) twist or an "horizontal" (H) twist.

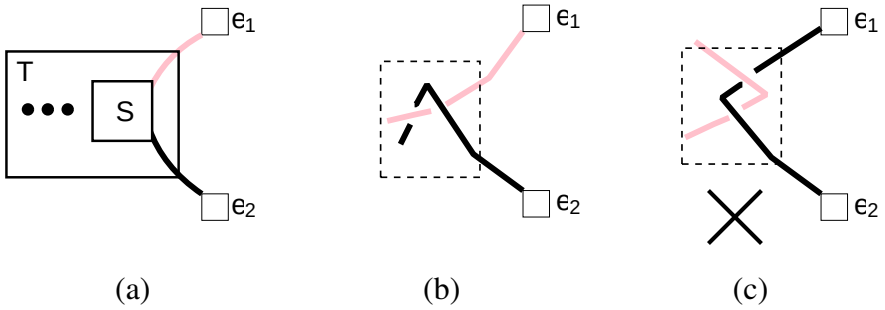
The untwisting can be done systematically by the following process. First, we put all the subtangles into a group defined by G. We start by finding the subtangle to be untwisted in G. The subtangle S that satisfies the following conditions is selected.

1. Two end points of S, defined here by e_1 and e_2, are also the end points of T (Figure 3.10(a)), which means there is no other subtangle between the end points of S and those of T.

2. S can be untangled by twisting e_1 and e_2

The second condition can be judged by checking whether (i) the closest minimal tangle from e_1 and that from e_2 are the same (Figure 3.10(b)). (Remember the minimal tangles

are those whose GLI values are above 0.5 which are found using the method explained in Section 3.2.3.1) and that (ii) e_1 and e_2 are not connected in this minimal tangle (Figure 3.10(c)).



(a) (b) (c)

Figure 3.10: The conditions for selecting the subtangle S to be untangled. (a) The two ends of S needs to be the end points of T. (b) The closest minimal tangle from e_1 and e_2 must be the same (c) e_1 and e_2 is not connected in this minimal tangle.

Once the subtangle S that satisfies these conditions is found, we virtually untangle this subtangle by removing it from G, and putting it into the top of *TangleList* and repeat the process iteratively until the whole tangle is untangled. When untangling a subtangle, we record its type (either of *BB*, *AB*, *BA*, *AA*).

For tangle types AB and BA, the tangles can be efficiently detected using the method explained in Section 3.2.3.1. For the tangle types AA and BB, the tangles were composed by the rigid segments from the same strand. In these cases, the writhe matrices are different from those in type AB and BA. Here we prepare another writhe matrix, which is computed by calculating the GLI between the rigid segments on the same strand, for checking tangle types AA and BB. As a result, the system scans 3 matrices - one for type AB or BA, another for type AA and the last one for type BB for detecting and encoding the tangles.

When checking the tangle type, the system first checks the GLI matrices computed from the two paths, $path_1$ and $path_2$. Since type AA and BB are self-tangles and have a single direction only, the system can recognize them by checking the GLI matrices computed from $path_1$ against $path_1$ and $path_2$ against $path_2$.

For type AB and BA, the difference between them is the directions of the paths as shown in Figure 3.9. In Figure 3.11, the elements with large GLI value were shaded in black. For rational tangles, the dark areas tend to form a straight line. From this observation, we found that we can estimate the directions of the path by check the slope of the straight line formed by the dark areas. If the slope is negative, it is a

AB-type tangle. If the slope is positive, it is a BA-type tangle.

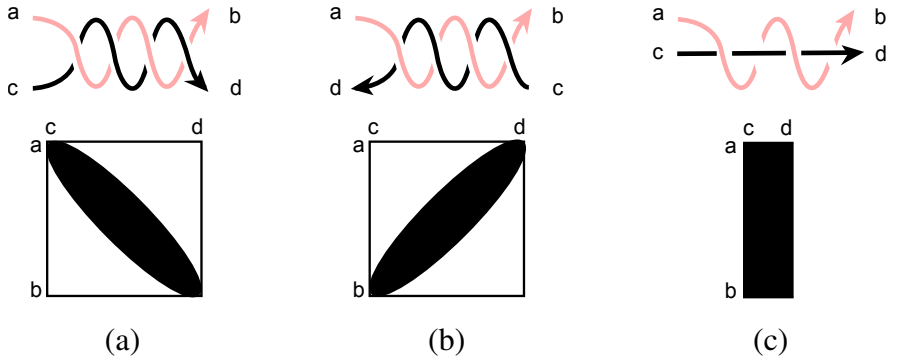


Figure 3.11: Example of rational tangles and their corresponding GLI matrices.

There is a special case which you cannot compute the slope for the straight line, an example is shown in Figure 3.11 (c). If a single long segment is tangled with a chain, a $n \times 1$ matrix will be computed where n is the number of segments on the chain. In this case, we could not estimate the direction of the paths. Our system handles this special case by checking the Euclidean distances between the end-points of the segments. By doing this, we can estimate the direction of the paths. For human postures/motions, however, it is less likely to have a single long segment tangled with other body segments.

Once the type of the tangle has been detected, the system compares the types of consecutive tangles. If the type of the tangle is different from that of the previous subtangle, we can say that the twist direction has switched too. Otherwise, the consecutive subtangles with the same type will be merged. The merging operation is the same as concatenating the rational tangles.

If no subtangle that satisfies condition (1) and (2) is found, T is not a rational tangle. In that case, we do not use this tangle to evaluate the similarity. The process of encoding the tangle information is summarized in Algorithm 1.

Two examples of tangles encoded by this method are shown in Figure 3.12. Whether the two tangles are the same or not can be checked by comparing their *TangleLists*. The details are explained in the next subsection.

3.2.5 Computing Similarities by Topological Relationships

Given two different sets of postures where the two characters are tangled with another, we compute the similarities of the postures / movements by using the encoded tangle information.

Algorithm 1 Encoding the tangle information

Put all subtangles in G
while G is not empty **do**
  Pick out the subtangle S in G that satisfies condition 1 and 2
  if $S = NIL$ **then**
    T is not a rational tangle. Exit()
  end if
  Check the two ends of S and set $S.type$.
  if $TangleList$ is empty **then**
    $S.twist = H$
  else
    if $S.type != S_p.type$ **then**
      $S.twist = !S_p.twist$
    else
      merge S and S_p and define it as S
      $S.twist = S_p.twist$
      $S.type = S_p.type$
    end if
  end if
  $S_p = S$
  Add S into $TangleList$
end while

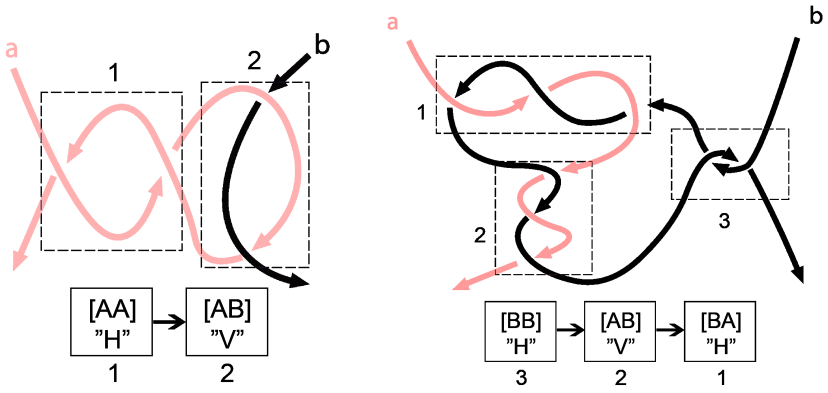


Figure 3.12: Examples of encoding rational tangles. The encoded TangleList is shown on the bottom.

As shown in Figure 3.4, there are 10 paths connecting every pair of end-effectors in the articulated structure we use in the system. Therefore, there are $10 \times 10 = 100$ pairs of paths formed between two characters. For every pair of paths, the system builds the TangleList structure, in which all the subtangle information is saved.

In order to compare the similarity between the pairs of postures, we have prepared two distance functions according to the application. Suppose we have two pairs of postures: the first pair is defined by u and the other by v, and there are m paths on u and n paths on v, respectively. Each pair of postures has $m \times n$ TangleLists, each of which represents how the paths between the end effectors are tangled.

In the first distance function, we evaluate the overall topological similarity between two pose pairs, which will be useful for managing a database of human motions. This is done by accumulating the distance between the corresponding TangleLists in u and v:

$$d = \sum_{i=1}^{m} \sum_{j=1}^{n} dist(TangleList_i^u, TangleList_j^v) \tag{3.1}$$

where i and j are the indices of the paths, and $TangleList_i^u$ and $TangleList_j^v$ are the i-th TangleList in u and j-th TangleList in v, respectively, and $dist()$ is a function that computes the distance between two TangleLists.

The distance between TangleLists is computed by finding the matching subtangles in the two TangleLists and comparing their GLI values. When matching a subtangle, we use a method similar to dynamic time warping: if the l-th subtangle of TangleList 1 is matched with the k-th subtangle of TangleList 2, subtangles of TangleList 1 whose index is larger than the l cannot be matched with subtangles of TangleList 2 whose

index is smaller than k.

$$dist(TangleList_i^u, TangleList_i^v) = \min_{j_l^u, j_l^v} \sum_{k=1} (TangleList_i^u[j_k^u].GLI - TangleList_i^v[j_k^v].GLI)^2 + P \quad (3.2)$$

where $0 < j_1^u < ... < n_{u,i}$, $0 < j_1^v < ... < n_{v,i}$ and $n_{u,i}$ and $n_{v,i}$ are the number of subtangles in $TangleList_i^u$ and $TangleList_i^v$, respectively, and P is the sum of squares of the GLI of the subtangles in $TangleList^u$ and $TangleList^v$ which could not find a matching subtangle.

The second distance function is used to judge whether postures and motions can be interpolated or blended without inter-penetrations of the segments. For such application, it is more meaningful to evaluate the maximum GLI difference rather than accumulating the difference of GLI for all the combination of paths:

$$blendable(TangleList_i^u, TangleList_i^v) = \max_k (\min_{j_l^u, j_l^v} |TangleList_i^u[j_k^u].GLI - TangleList_i^v[j_k^v].GLI|). \quad (3.3)$$

We can estimate that when $blendable()$ returns a value larger then 0.5, it is unlikely that the postures can be linearly interpolated without penetration, as some parts of the body need to be tangled / untangled.

3.3 Experimental Results

The results of computing the similarities of posture pairs based on the topological relationships are presented in this section. We also show examples of interpolating / concatenating different motion clips by using topological relationships as a measure.

3.3.1 Comparison between Topological and Euclidean distance

We first compared the performance of the distance metrics of Eq. 3.2 which is based on topological relationship, and that based on Euclidean distance, to distinguish posture pairs which are semantically similar / dissimilar. Fifty eight posture pairs which can be divided into the following seven categories were prepared:

- Full Nelson holds (motion 1-8)
- back holds (9-11)

- firefighter carries (12-15)
- back-breakers (16-22)
- octopus holds (23-25)
- one person carries (26-32)
- piggyback-carries (33-36)
- walk support (37-39)
- dancing (40-49)
- Latin dance (50-58)

We computed the normalized similarities between the postures, whose values are between 0 (less similar) and 1 (highly similar) by using both the topological and Euclidean distance metrics. They are computed by $1 - d/d_{max}$, where d is the distance between the posture pairs and d_{max} is the maximum distance found among all the pair postures used in this research. For the Euclidean distance, we used the point cloud metric proposed by Kovar et al. [Kovar et al., 2002].

The results are visualized in Figure 3.13 and Figure 3.14. The darker areas represent higher similarity and the lighter areas represent lower similarity. The postures of the same kind are grouped together along the row/columns.

It can be observed that in the similarity matrix based on the topology, the distance between postures in the same group of actions are small. On the other hand, that does not necessarily apply to the results based on the Euclidean distance. For example, when using the topological distances, it can be observed that all the motions in the category of Full Nelson holds, back holds, and firefighter carries are evaluated as similar, while there are significant variations when using Euclidean distances.

In some categories, the topological relationships between the characters are different. Those postures are 20, 21, and 22 in back breakers, 25 in octopus holds, 28 to 32 in one person carries, and 36 in piggy back carries. Therefore, their topological distances from the other motions in the same category are relatively large. However, in some cases, the Euclidean distance between such postures with other postures in the same category are small because the positions of the joints are similar. This happens at posture 26 in octopus hold; although the Euclidean distance with the other postures are small, its topological relationship is actually different.

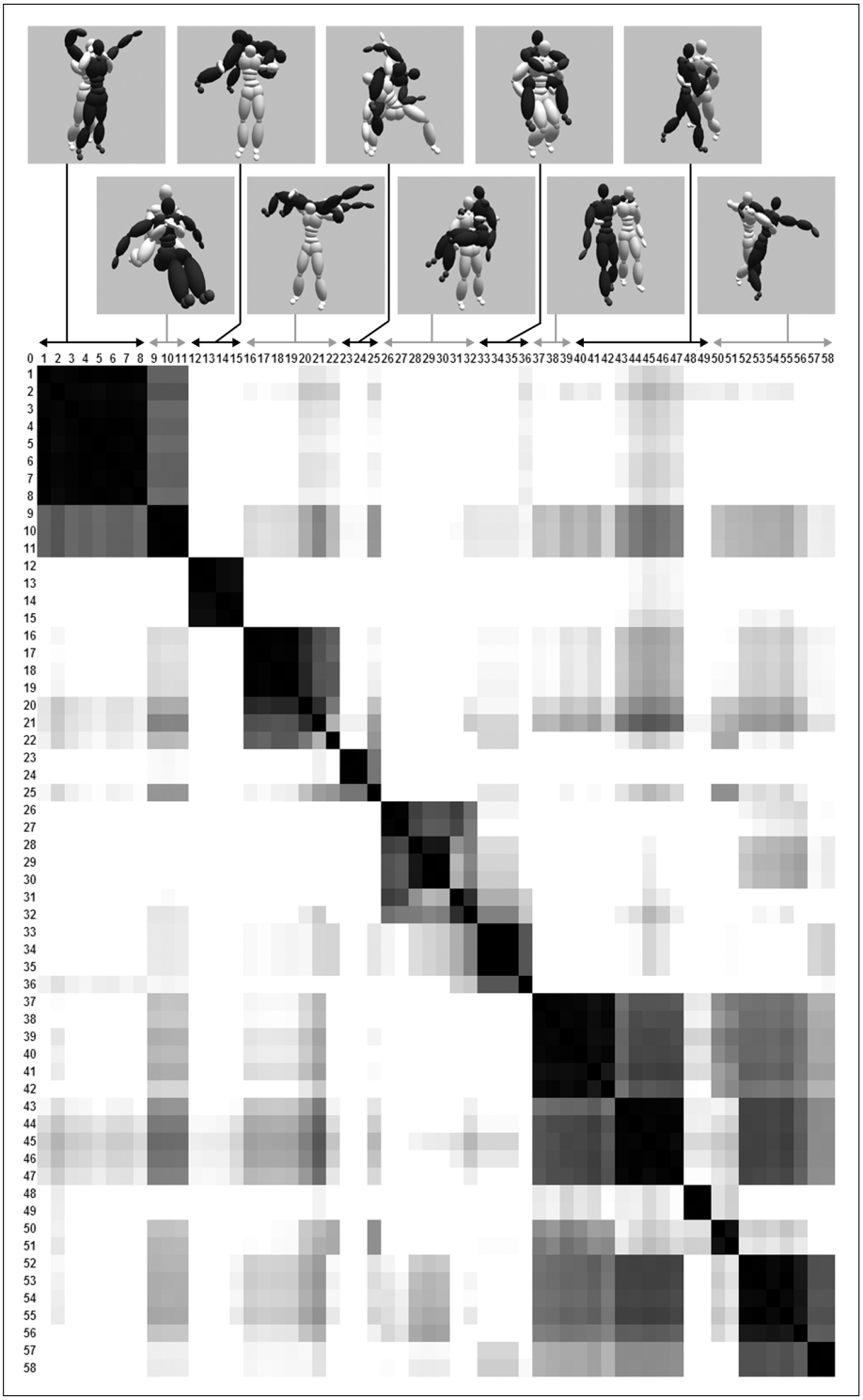


Figure 3.13: A similarity matrix of different postures based on topological relationships.

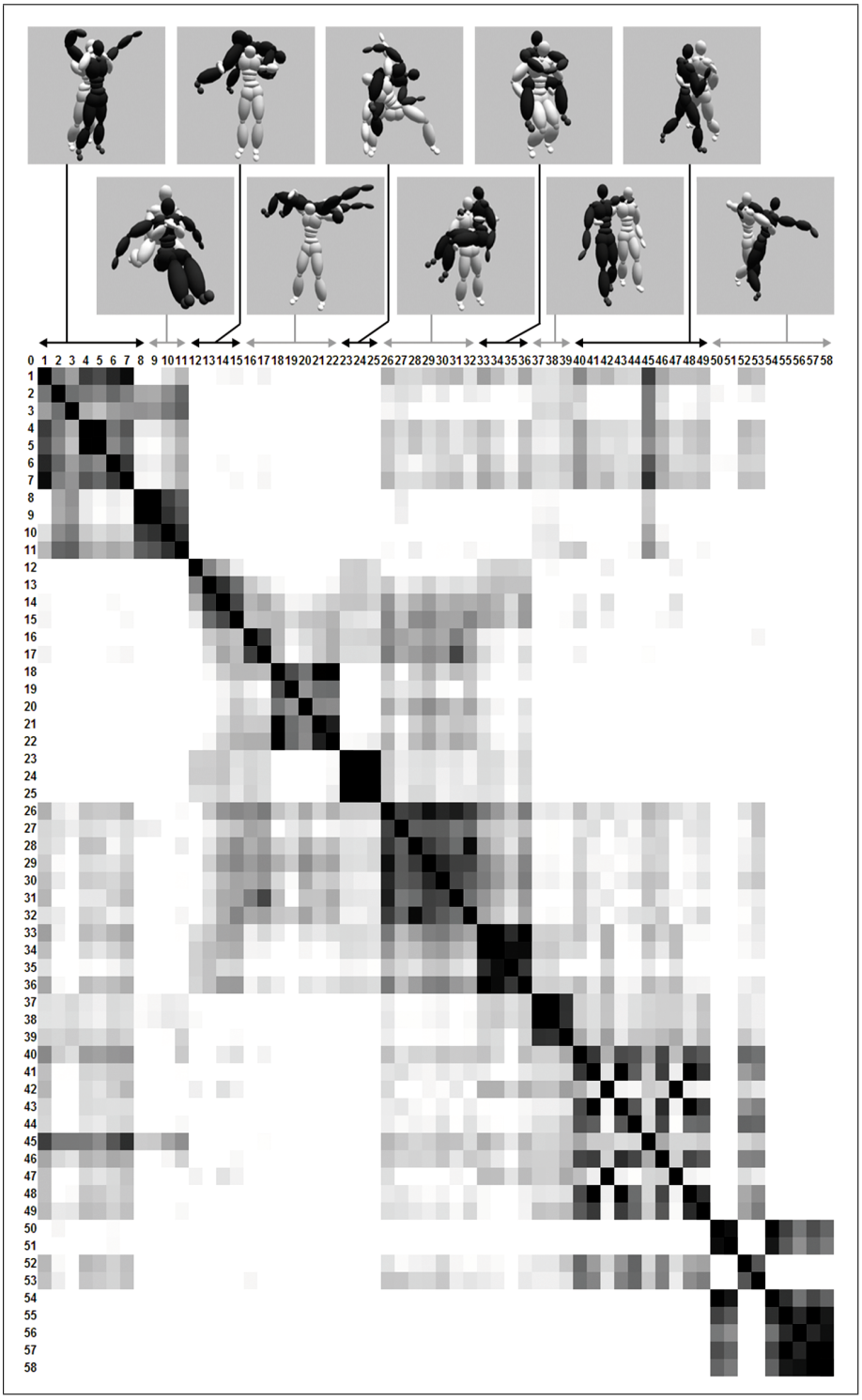
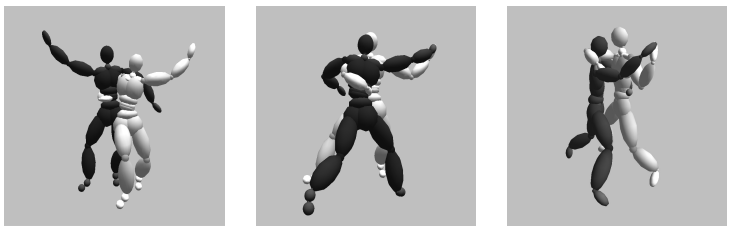


Figure 3.14: A similarity matrix of different postures based on Euclidean distance computed by Kovar et al.'s point cloud metric [Kovar et al., 2002].

The dancing postures (40-49) are composed of three groups. In the first group (40-42, Figure 3.15 (a)), the white character is holding the torso of the black character, and the black character is holding the neck of the white character. These postures are topologically equivalent to those of the walk assisting motion (37-39). In the second group (43-47, Figure 3.15 (b)), the white character is holding the torso of the other, and the third group of postures are same as those of the first group, but the role of the two characters are switched (43-47, Figure 3.15 (c)). The proposed method can differentiate the postures correctly as shown in Figure 3.13.

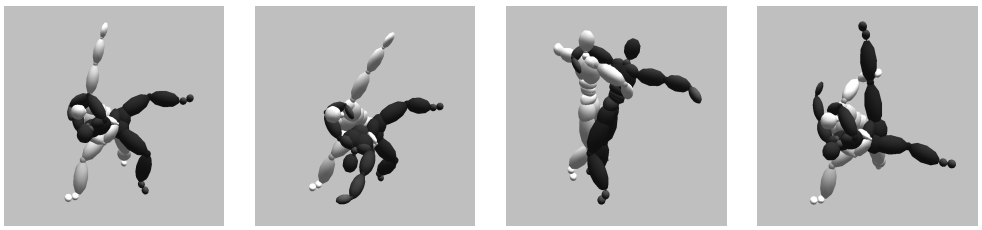
In all the Latin dance postures (50-58), the left arm of the white character is tangled with the torso of the black character. In addition to that, the black character's two arms (50, Figure 3.16(a)), left arm (51, Figure 3.16(b)) or right arm (52-58, Figure 3.16(c)) are tangled with the neck of the white character. In posture 57-58, there is an additional tangle between the white character's left arm and the black character's right leg (Figure 3.16(d)). Although the postures in the same group are topologically equivalent, they are kinematically dissimilar. It is difficult to classify them in the kinematic framework, as shown in Figure 3.14. Since the white character's right arm is tangled with the torso of the black character and the left arm of the black character is tangled with the neck of the white character in most of the dancing and Latin dance postures, the topological similarity of these postures are large, and as a result, a large gray area exists in the similarity matrix in Figure 3.13. Such topological similarity cannot be detected by Euclidean distance, and there is no consistency in the corresponding region of the similarity matrix shown in Figure 3.14.

In the similarity matrix computed by Euclidean distances, it can be observed that the variance of the similarity is large in both cases that the postures are in the same or different groups. The large variance within the same group implies that the similarity is largely dependent on the kinematics and can result in inconsistencies. There is a lot of risk that it treats semantically different postures as similar and semantically similar postures as different. On the other hand, in the similarity matrix computed by topology distance, the variance is low. In some cases, the topology distance returns false positive results. For example, a grey area exists in 50-57 x 38-47 of the topology distance-based matrix. This is simply because the postures in these groups are topologically similar. The low variance suggests the consistency within the same group, and little influence by the kinematics of the postures.



(a) (b) (c)

Figure 3.15: The representative postures of the three groups of dancing motions (42,43,48).



(a) (b) (c) (d)

Figure 3.16: The representative postures of the Latin dance motions (50,51,52,57).

3.3.2 Content-based retrieval

Next, an experiment of content-based retrieval was done. The example postures were given as queries and the results together with the numerical similarity of each pair were computed and returned to the user.

Some of the results are shown in Figure 3.2 and 3.17. The similarities of the postures are computed by Eq. 3.2. It can be observed, postures with similar topological relationships are at the top of the lists.

We further compare the performance of our proposed method with the traditional point-cloud approach for motion retrieval. Here we compute the precision, which is the ratio of relevant results to retrieval results, for the postures in the same database used in the experiments in Section 3.3.1. The precision-scope curve and precision rate are derived by averaging the results from queries using different full-nelson hold postures as input. The precision-scope graph (Figure 3.18) shows that our method returned relevant (i.e. topological equivalent) results as top-ranked results while the point cloud metric is failing to distinguish the postures with different topological relationships. The ambiguity in differentiating the postures may lead to a lot of collisions and inter-penetrations of the body segments if we use the retrieved postures/motions for motion blending or concatenation.

Figure 3.17: Results of content-based retrieval based on the topological relationships. Despite the large variation of the postures, the pairs of postures with similar topological relationships return high scores.

3.3.3 Creating animations by Motion Graph

Thirdly, Motion Graphs [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002] were created based on two sets of posture / motion data - one set on wrestling (**Group A**, Figure 3.19) and the other on dancing (**Group B**, Figure 3.20) - and we evaluated the animations created based on these Motion Graphs.

In Motion Graphs, the nodes represent postures and edges represent transition motions between the postures. They can be produced by comparing the distance between every posture of the captured motion data and connecting nodes by edges whose distances are below a given threshold. Here we created a small-scale Motion Graph using the postures in Section 3.3.1 and some additional short motion clips. We calculated the distance between all the postures and also the initial / final posture of the short motion clips, and connected the postures by edges if the distance was smaller than a threshold.

Three different methods were used to create the graphs and the results were compared. Firstly, we used the point-cloud distance metric in [Kovar et al., 2002] to compose the graph. Secondly, we used the topological distance of Eq. 3.3 explained in Section 3.2.5 to compose the graph. This is because we want to find out whether the two postures can be blended or not rather than finding out the overall topological similarity of the two posture pairs. We had tightened the threshold to 0.45, which is slightly below 0.5 to reduce the risk of inter-penetrations of the segments. Finally, we used a

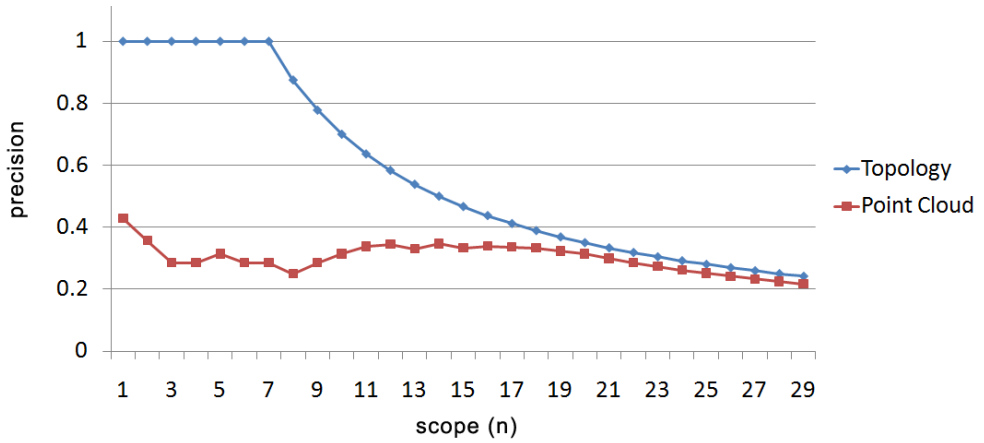
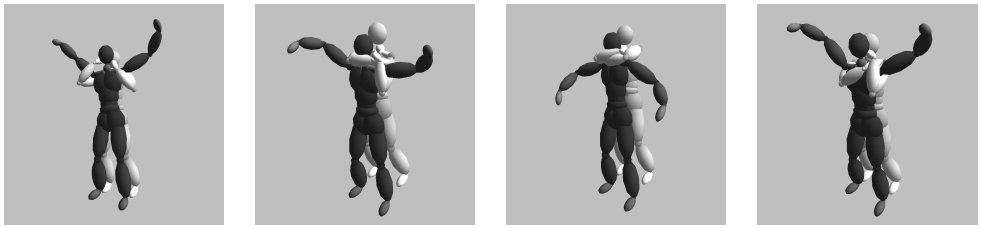


Figure 3.18: Precision-scope curves showing the performance of our proposed method and the point cloud metric.

2-pass method which first filters out postures whose topological relationships are too different and then compares the closeness based on the point-cloud metric.

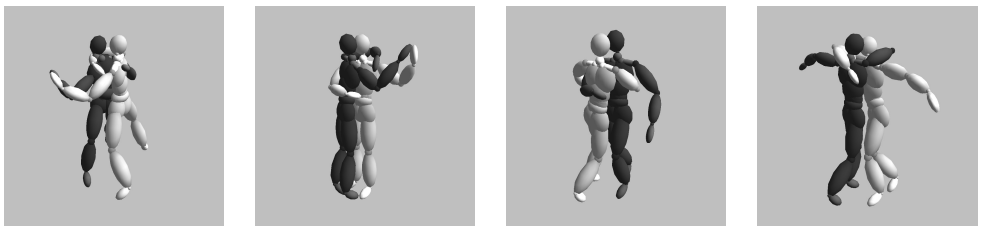


(a) (b) (c) (d)

Figure 3.19: The Full-nelson hold and Rear Chokehold postures/motions in Group A.

The statistics of the Motion Graphs created based on the postures/motions of Group A and B are shown in Table 3.1 and 3.2, respectively. In the tables, *invalid edges* refer to the edges (transition motions) causing the bodies to intersect/penetrate each other. By using the point cloud metric alone, a lot of invalid edges were found as many postures within each group are numerically similar, although their topological relationships are different. Those apply to postures shown in Figure 3.19(b) and 3.19(d), for example.

By using the topological distance alone, the number of invalid edges is reduced significantly. Since only topological distance is compared, some of the edges are connecting two numerically dissimilar postures. We found that most of these edges are valid edges. However, blending numerically dissimilar postures is not preferred in Motion Graphs since discontinuous motions will be created. The risk of intersecting



(a) (b) (c) (d)

Figure 3.20: The dancing postures/motions in Group B.

/ penetrating will also rise as the bodies need to move a lot when interpolating the postures.

By using the 2-pass method, we only connected the postures whose point-cloud and topological distances are both small; as a result, no invalid edges are found. This implies that taking into account the topological distance between the postures is very useful in motion blending and concatenation, especially when the motion database contains a lot of motions whose topology are different.

	Euclidean	Topological	2-pass
# of edges	98	110	28
# of invalid edges	50	8	0

Table 3.1: Statistics of the Motion Graph based on wrestling postures / motions

	Euclidean	Topological	2-pass
# of edges	120	110	48
# of invalid edges	47	18	0

Table 3.2: Statistics of the Motion Graph based on the dancing postures / motions

3.3.4 Creating animations by concatenating motion clips

Finally, we used topological distance as a measure to evaluate whether motion clips can be concatenated or not. Three motion clips were prepared: (a) a person giving a shoulder to assist the walk of another (Figure 3.21,top), (b) a person doing a one-person arm carry to another (Figure 3.21, middle), and (c) a person conducting a back drop to another (Figure 3.21, bottom). First, we computed the *TangleLists* of the

two bodies at every frame of the motion clips. Based on the topological relationship, the walking motion and the carrying motion are divided into three stages, and the backdrop motion is divided into two stages. By comparing the *TangleLists* of every frame between the motions, it was found that the topological relationship at the third stage of the walking motion and at the first stage of the carry and backdrop motion are the same. We have further found the best frame to concatenate the walking motion and the carry motion, and the walking motion and the backdrop motion. By comparing the Euclidean distance of the joints, the smooth transitions from the walking motion to the carry motion and the walking motion to the backdrop motion were achieved.

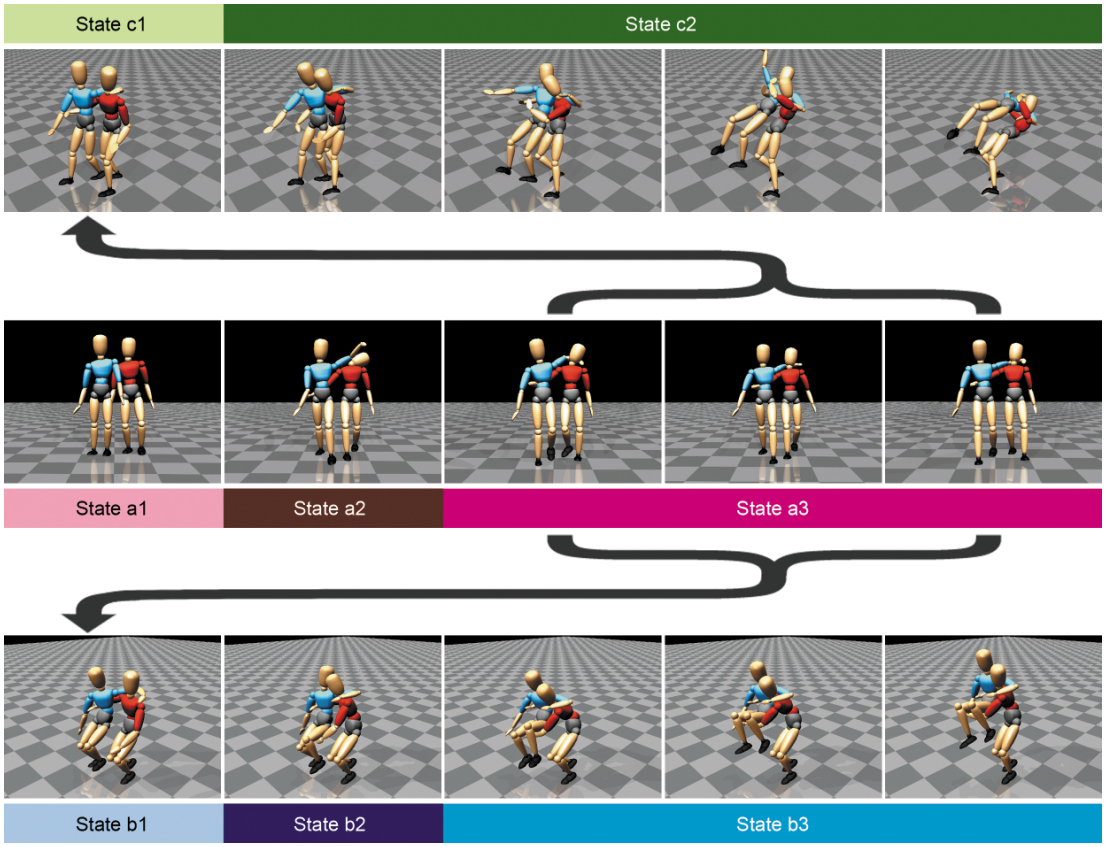
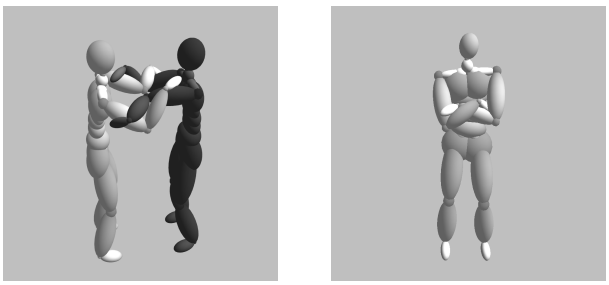


Figure 3.21: The matching stages of the assist-walking motion, the backdrop motion and the lifting motion.

3.4 Discussions and Conclusion

In this chapter, we have proposed a new method to index postures of two characters closely interacting with each other. The method is based on the theory of rational tangles, and it is shown that we can categorize various postures of two characters tangled



(a) (b)

Figure 3.22: Postures with: (a) prime tangle (b) self tangle.

with each other. We have also shown that a baseline method using low level attributes such as the position of the joints can suffer from categorizing such postures.

We have limited the tangles made between the segments to rational tangles. There are also different categories of tangles called self-knotted tangles and prime tangles. Such tangles have more complex structures and there is still no invariant known for them. However, since we are limiting our subjects to human bodies, we do not have to worry about them for the following reasons: (1) The human body is composed of a limited number of rigid segments, and therefore, it is difficult to compose self-knotted / prime tangles by the human body. (2) Although self-knotted / prime tangle are composed at some paths, usually there are other rational tangles sharing the subtangles with those self-knotted / prime tangles. Our system then encodes such rational tangles and use them to distinguish the postures. As a result, by simply excluding the self-knotted / prime tangles from considerations, we can compute the similarities of the pairs of postures by using the TangleLists of the rational tangles.

In the experimental results, we have shown several examples concatenating different motion clips by using Motion Graphs [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002]. By comparing different distance metrics, the results clearly showed that using topological distance as a measure can reduce the number of collisions or penetrations in the blended motions. However, there are problems when interpolating topologically similar but kinematically dissimilar postures. For example, the location of the supporting feet can be quite different, which makes the resulting motions discontinuous and unnatural. By combining the topological and Euclidean distance metrics, the blended motions are free from collisions or penetrations while the visual quality (in terms of continuity of the motions) is comparable to those created by conventional Motion Graphs. Taking into account the topological distance helps to generate collision-free motions automatically, especially when the motion database

contains motions in which multiple characters closely interact with each other.

We mainly conducted experiments at the posture level instead of the motion level. It is straightforward to extend this concept to the motion level where the topological relationships change over time. As explained in Section 3.3.4, we can segment the motion at the postures when the topological relationship changes, and index or retrieve the motions using a sequence of topological relationships.

We proposed to encode the tangles made between the global paths connecting the end effectors; another approach to encode the tangles is to compute the local GLI between shorter paths such as those made by the limbs. Such an approach might be more efficient as we will only need to encode the local area where the tangles are composed. However, a drawback is that another approach to estimate the similarities of postures which are composed of different segments will be required. For example, the two postures in Figure 3.5 are composed of different segments, although they should be considered similar. As our method is based on the topological relationship of global paths, such postures are treated as equivalent. We can further compare the details by comparing the kinematical difference.

Chapter 4

Character Motion Synthesis by Topology Coordinates

Having presented a systematic way to index postures of two characters closely interacting with each other by topological representation in Chapter 3, we now present how the topological information can be used in character motion synthesis. We introduce the concept of Topology Coordinates, in which the topological relationships of the segments are embedded into the attributes. As a result, the computation for collision avoidance can be greatly reduced for complex motions that require tangling the segments of the body. Our method can be combinedly used with other prevalent frame-based optimization techniques such as inverse kinematics.

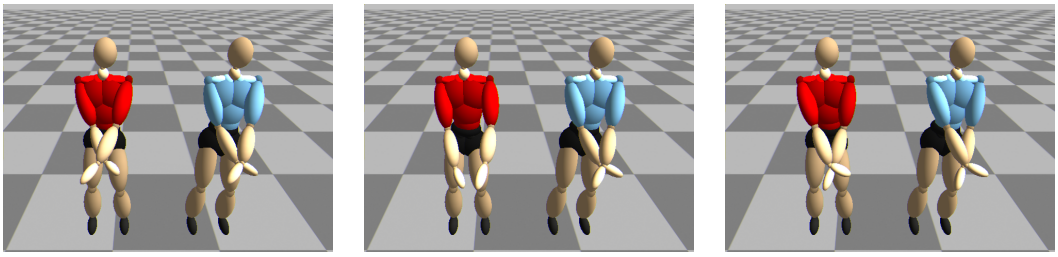
Portions of this chapter have previously been published as [Ho and Komura, 2007a], [Ho and Komura, 2009a], [Ho and Komura, 2009c] and [Ho and Komura, 2010].

4.1 Introduction

Despite the huge amount of research that has been done in the field of robotics and computer animation, synthesizing motions that involve close contacts is still a challenging task. Previous research suffers from a huge amount of computation for collision avoidance as path-planning is done at the level of generalized coordinates or Cartesian coordinates, which are the lowest level of state representation.

In this research, we efficiently plan such complex motions by introducing the idea of *Topology Coordinates* which takes into account the topological relationships of the segments. In Topology Coordinates, *writhe*, which represents how much the segments twist around each other, is the main attribute of the state space. We can easily avoid

collisions of segments by simply moving the segments along the axis of writhes in Topology Coordinates. For the rest of this article, let us call the manifold represented by the Topology Coordinates as *Topology Space*.



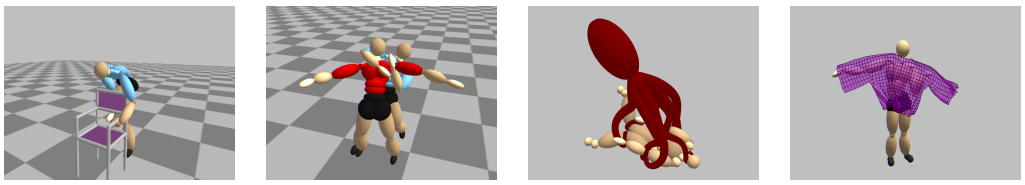
(a) (b) (c)

Figure 4.1: The snapshots of interpolating the postures in (a) and (c) by Topology Coordinates (left, red character) and by generalized coordinates (right, blue character). Postures in (b) are the intermediate postures. The two arms twist around each other when they are interpolated by Topology Coordinates, while they penetrate through each other when they are interpolated by generalized coordinates.

Let us think of the example of interpolating the two postures shown in Figure 4.1 (a) and (c) again. The main difference of the two postures is that in (a), the left arm is crossing over the right arm, and vice-versa in (c). Although the two postures are similar at the level of generalized coordinates, the arms will penetrate through each other if they are linearly interpolated, as the blue character is doing. On the other hand, the two postures are far apart from each other in Topology Space. If they are linearly interpolated in Topology Space, we will obtain a motion in which the character twists its arms around each other, as the red character is doing.

We propose a method to interactively synthesize motions of close contacts by modeling the movements in Topology Space. The movements in Topology Coordinates can be easily mapped to / inversely mapped from generalized coordinates. Therefore, we can combine our method with keyframe animation and frame-based optimization.

Our approach is most suitable for creating motions which involve close contacts between characters / deformable objects. The user can interpolate keyframe postures or interactively control the Topology Coordinates to produce animations such as a character tangling its arms with a bulky furniture to hold it (Figure 4.2 (a)), two characters playing wrestling by switching from one tangled posture to another (Figure 4.2 (b)), an octopus tangling its limbs with a human character while avoiding its limbs getting tangled themselves (Figure 4.2 (c)) and a human character wearing a T-shirt (Figure



(a) (b) (c) (d)

Figure 4.2: Motions that involve close contacts are generated by controlling the characters in Topology Space. The user can interpolate keyframe postures or interactively control the Topology Coordinates of characters to produce animations such as (a) a character tangling its arms with a bulky furniture to hold it, (b) two characters playing wrestling by switching from one tangled posture to another, (c) an octopus tangling its limbs with a human character while avoiding its limbs getting tangled themselves and (d) a human character wearing a T-shirt.

4.2 (d)).

Our Approach: Instead of planning the motions at the level of generalized coordinates, we first plan the movements in Topology Space. The changes in the Topology Coordinates are mapped to generalized coordinates through optimization based on quadratic programming, which can handle other common constraints such as kinematic or dynamical constraints. Our method can be integrated with other frame-based optimization approaches, so that it can enhance the functions of previous methods.

The outline of our method to synthesize character animation is shown in Figure 4.3. The details of each step are as follows: (1) The user specifies how the characters tangle their bodies during the animation. The user specifies the segments of the body to be tangled using our user interface, and produces a keyframe posture by changing the Topology Coordinates. The concept of Topology Coordinates is explained in Section 4.2. Our interface to edit keyframe postures is explained in Section 4.4.1. (2) The animation of the characters is produced by interpolating the keyframe postures in Topology Space. The basic idea to control multi-segment chains by changing the Topology Coordinates is explained in Section 4.3. (3) If the user is not satisfied with the animation, he/she can further add kinematic constraints or control either of the characters interactively using inverse kinematics, or even replace the motion of character with captured motion data. The topological relationship between the characters can be kept by using Topology Coordinates as constraints. Examples of such animations are shown in Section 4.4.2.

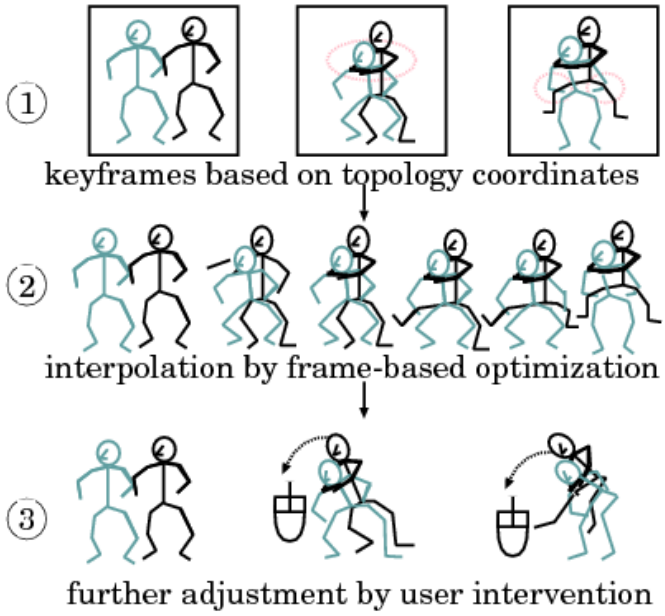


Figure 4.3: Overview: The keyframe postures are given based on the Topology Coordinates. The keyframes are interpolated by frame-based optimization. The user can further update the motions by dragging or constraining segments.

4.1.1 Contribution

- We propose a new state space called Topology Space, in which the topological relationship of the body segments are embedded in the coordinate system. As a result, motion synthesis of complex interactions that requires collision avoidance can be done efficiently.
- We propose a method to map the movements in Topology Coordinates to those in generalized coordinates.

4.2 Topology Space and Coordinates

In this section, we explain the fundamental ideas of topology space in which the topological relationships of multibody segments are embedded in the coordinate system. We also present their mathematical definitions.

In Topology Space, we assume the segments are modelled by curves or line segments. If the character has a multibody structure, we control the hierarchical bone structure of the characters in Topology Space.

4.2.1 Topology Coordinates

Here the three attributes of Topology Coordinates are explained. The first attribute is the **writhe**, which counts how much the two curves are twisting around each other. Writhe can be calculated by using Gauss Linking Integral (GLI) [Pohl, 1968]. The GLI of two directed curve can be calculated by Eq. 2.3.

Curves can twist around each other in various ways. In order to further specify the status of the two chains, we introduce two other attributes, **center** and **density**. Examples of changing these attributes for a pair of strands are shown in Figure 4.4. Center, which is composed of two scalar parameters, explains the center location of the twisted area. Density, which is a single scalar parameter, explains how much the twisted area is concentrated at one location along the strands. When the density is zero, the twist is spread out all over the two strands. When the density value is either very large or very small, we can say one strand is playing a major role to compose the twist, as it is twisting around the other strand which is kept relatively straight (Figure 4.4). When the density turns from negative to positive, or vice-versa, the strand playing the major role switches.

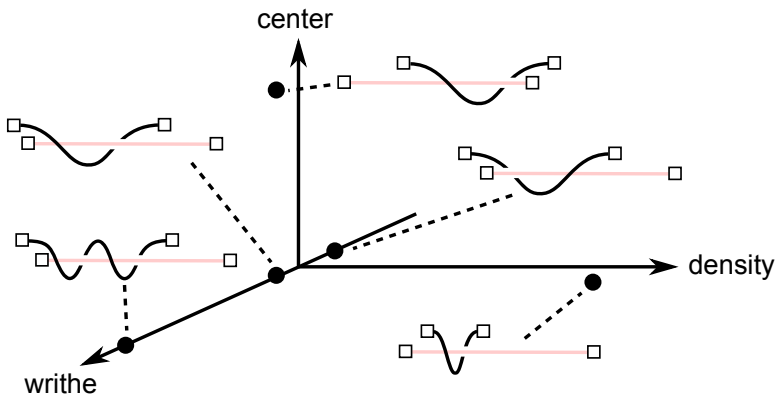


Figure 4.4: The three axes in Topology Space : writhe, center and density. Center, which specifies the central location of the twist, is composed of two scalar parameters although it is represented by a single axis in this figure. Density tells which strand plays the major role to compose the twist.

4.2.2 Mathematical Definition

We represent the bone structure of characters by a set of line segments. Therefore, now we will mathematically define the Topology Coordinates of serial chains. Let us assume we have two chains S_1 and S_2, each composed of n_1 and n_2 line segments,

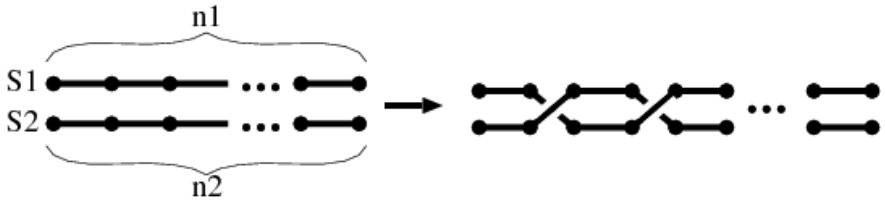


Figure 4.5: Twisting a chain of segments around each other

connected by revolute, universal or gimbal joints (Figure 4.5). In this case, we can compute the total writhe by summing the writhes by each pair of segments:

$$w = GLI(S_1, S_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} T_{i,j} \tag{4.1}$$

where w represents the writhe, $T_{i,j}$ is the writhe between segment i on $S1$ and j on $S2$. An analytical solution exists for computing $T_{i,j}$ [Levitt, 1983] and the details are

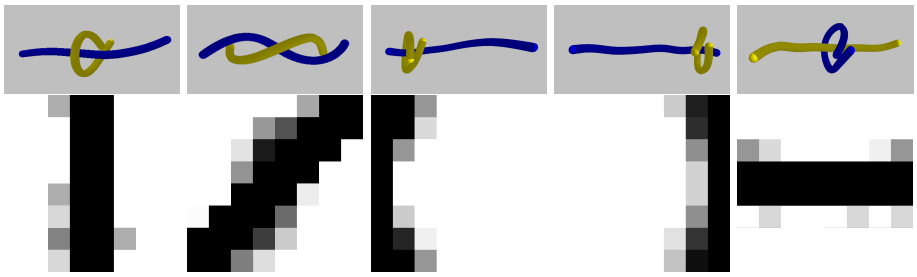


Figure 4.6: (upper) Tangles with different density and center, and (lower) the distribution of elements with large absolute values in the corresponding writhe matrix. The darkness represents the amplitude of the absolute value.

explained in Section 2.4.2 in the Related Work. Let us define a $n_1 \times n_2$ matrix $\mathbf{T}$ whose (i, j)-th element is $T_{i,j}$, and call this the **writhe matrix**. The writhe matrix explains how much each pair of segments from S_1 and S_2 contributes to the total writhe value. Various twists and the corresponding writhe matrix are shown in Figure 4.6.

We first define a 2D vector $\mathbf{c}$ as the center calculated by the following equation:

$$\mathbf{c} = (x_g, y_g) = \left(\frac{\sum_i^{n_1} \sum_j^{n_2} i \cdot T_{i,j}}{w} - \frac{n_2}{2}, \frac{\sum_i^{n_1} \sum_j^{n_2} j \cdot T_{i,j}}{w} - \frac{n_1}{2}\right) \tag{4.2}$$

This explains the center of the twisted chains in Topology Space. It tells the overall location of the twisted area.

In Figure 4.6, it can be observed that the elements with large absolute values concentrate along a particular axis across the matrix. This axis tends to be vertical when S1 twists around S2 and horizontal when S2 twists around S1. When both chains twist

around each other, the axis lies along the diagonal line. This observation leads us to the definition of the density to be the orientation of the principal axis. As different pair of chains result in different sizes of writhe matrices, we first normalize the data by scaling the writhe matrix to a square area, and then compute the principal axis in this area. We can compute the orientation of this axis against the diagonal line and define this as the density (Figure 4.7). Although the density is defined as such, we do not actually need to compute the density value from an existing writhe matrix. This definition is just convenient as we update the density value through rotating the elements within the writhe matrix. This process is explained in Section 4.3.

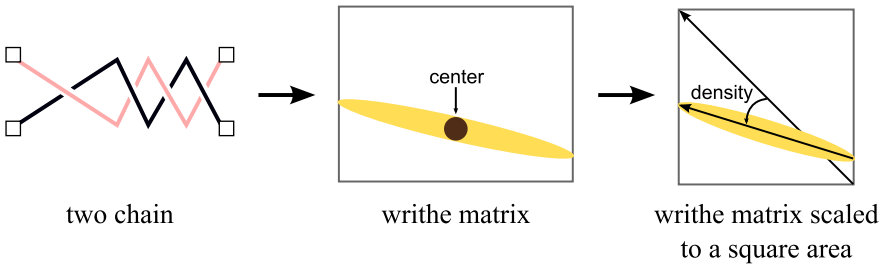


Figure 4.7: A twisted chain (left) and the writhe matrix (middle). The center is computed by calculating the center of mass of the writhe matrix's elements. The writhe matrix is scaled to a square area (right) to compute the principal axis. The angle made between the principal axis and the diagonal line is defined as the density.

We have explained how to compute the Topology Coordinates from the kinematics of the chains in this section. We can also solve the inverse problem; computing the kinematics from the Topology Coordinates. The methodology of such a manipulation, which is useful for generating motions that involve twisting, is explained in the following section.

4.3 Manipulation in Topology Space

In this section, we first explain how to synthesize the motions of a pair of serial chains, composed of n_1 and n_2 segments, by changing their Topology Coordinates. The Topology Coordinates of the chains are gradually updated and their movements are mapped to the generalized coordinates (Figure 4.8). This process is repeated until the Topology Coordinates reach their target values. At every step, suppose we want to change the Topology Coordinates from $\mathbf{G} = (w, d, \mathbf{c})$ to $\mathbf{G} + \Delta\mathbf{G} = (w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$. Using the updated Topology Coordinates, a corresponding writhe matrix is computed (step 1

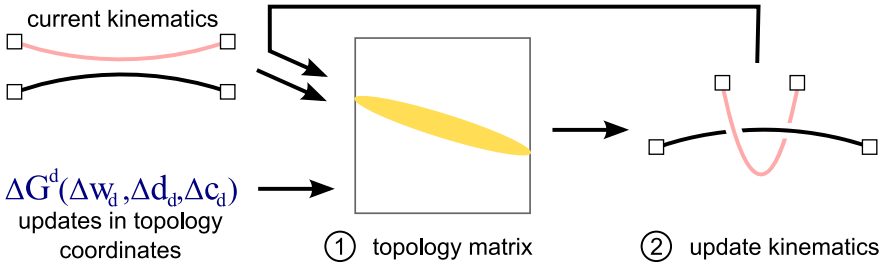


Figure 4.8: The overview of updating the kinematics of two serial chains by changing their Topology Coordinates. The increment/decrement of the Topology Coordinate is given at each step. The writhe matrix that corresponds to the updated Topology Coordinate is computed (step 1), and then the kinematics of the chains are updated so that their writhe matrix becomes similar to the target one (step 2). This process is repeated until the target Topology Coordinate is reached.

in Figure 4.8). This process is explained in Section 4.3.1. Then the kinematics of the serial chains are adjusted so that their writhe matrix becomes similar to the one computed from the Topology Coordinates (step 2 in Figure 4.8). This process is explained in Section 4.3.2.

Finally, we explain how this methodology can be used for two other manipulations: one is to pass a chain through a loop, and the other is to tangle a chain with a bundle of chains. These methods are explained in Section 4.3.3.

4.3.1 Desired Writhe Matrix

Here we explain a method to compute a writhe matrix that corresponds to a given Topology Coordinates coordinate $G = (w^d, d^d, \mathbf{c^d})$. This is done by first preparing a matrix which has all the values concentrated at one or two columns of the matrix, and then rotating and translating this part inside the matrix.

We start from a $n_1 \times n_2$ matrix $\mathbf{I}$, who has values evenly distributed at the $(n_2+1)/2$-th column if n_2 is odd, or at both the $n_2/2$ and $n_2/2+1$-th column if it is even:

$$\mathbf{I} = \begin{pmatrix} 0\cdots,\frac{1}{n_1},\cdots,0 \\ \vdots \\ 0\cdots,\frac{1}{n_1},\cdots,0 \end{pmatrix}, \qquad \begin{pmatrix} 0\cdots,\frac{1}{2n_1},\frac{1}{2n_1},\cdots,0 \\ \vdots \\ 0\cdots,\frac{1}{2n_1},\frac{1}{2n_1},\cdots,0 \end{pmatrix}$$

(n_2 is odd) (n_2 is even)

Note that for this writhe matrix, the corresponding Topology Coordinates are $w = 1, d = -\frac{\pi}{4}, \mathbf{c} = (0,0)$. In order to obtain a writhe matrix for arbitrary Topology Coor-

dinates, we rotate and translate the elements of $\mathbf{I}$.

4.3.1.1 Updating the density

As the density was defined as the orientation of the main axis of the writhe matrix, rotating $\mathbf{I}$ around the center results in changing the density. We first map the elements of the writhe matrix to a circle, rotate the values around the center until the orientation reaches the target density value d^d, and finally inverse-map the values onto the matrix (Figure 4.9). The mapping from the matrix to the circle is done by first mapping the matrix into a square area, and then into a circle area. The inverse mapping is done vice-versa. This operation is defined as $\mathbf{M}' = R(\mathbf{I}, d^d)$, where $\mathbf{M}'$ is the output matrix. The details of this operation are explained below:

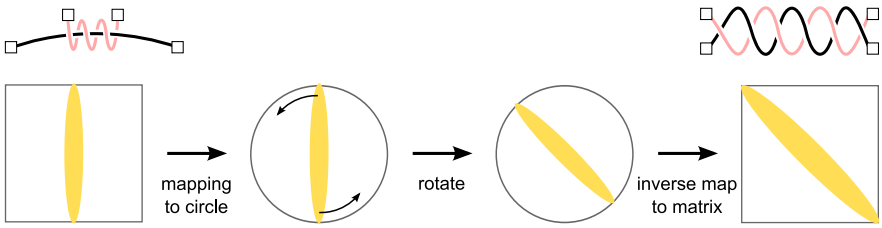


Figure 4.9: The elements of the writhe matrix are first mapped to a square area, and then to a circle. The circle is rotated until the axis reaches the desired density value. Finally, the axis is mapped back to the writhe matrix.

- Firstly, the elements of the input matrix M are mapped to an area within a square which is centered at the origin of a 2D Cartesian coordinate, and whose edges are over $x = \pm 1$ and $y = \pm 1$. This is done by

$$v_i = \frac{i - o_i}{o_i}, v_j = \frac{j - o_j}{o_j}.$$

  where (o_i, o_j) is the center of the input writhe matrix

$$o_i = \frac{n_1 + 1}{2}, o_j = \frac{n_2 + 1}{2}.$$

- Secondly, (v_i, v_j) are mapped into an area inside a circle whose radius is 1 by the following equation:

$$v'_i = v_i \sqrt{1 - \frac{v_j^2}{2}}$$

$$v'_j = v_j \sqrt{1 - \frac{v_i^2}{2}}$$

- Thirdly, (v_i'', v_j'') is rotated ϕ units around the origin according to the target density value d^d.

$$(v_i'', v_j'') = R(\phi)(v_i', v_j').$$

- Finally, (v_i'', v_j'') is inverse-mapped back into the square area, and the indexes in the new writhe matrix are computed:

$$v_i''' = \frac{v_i''}{\sqrt{1 - \frac{v_j''^2}{2}}}$$
$$v_j''' = \frac{v_j''}{\sqrt{1 - \frac{v_i''^2}{2}}}$$
$$i' = o_i + o_i v_i'''$$
$$j' = o_j + o_j v_j'''$$

  where (v_i''', v_j''') is the vertex in the square area and (i', j') are the indices where the elements of $\mathbf{M}$ at indices (i, j) are mapped to. As (i, j) went through transformations of scaling and rotation, (i', j') are no longer integers. Therefore, at the last stage, we will compute the elements of the new writhe matrix $\mathbf{M}'$ by using the weighted sum of entries at (i', j'):

$$\mathbf{M}'(i'', j'') = \frac{\sum_{k=1}^{4} D(i'' - i_k', j'' - j_k')\mathbf{M}(i, j)}{\sum_{k=1}^{4} D(i'' - i_k', j'' - j_k')}$$

  where (i'', j'') are the new integer indices of matrix $\mathbf{M}'$, $(i_k', j_k')(k = 1, ..., 4)$ are the transformed indices of the four neighbouring elements (located at upper left, upper right, lower left and lower right) of (i'', j'') and $D()$ represents the Euclidean distance operator.

This operation of rotation results in changing the density of the chain-pairs.

4.3.1.2 Updating the center

For changing the center, we simply translate all the elements according to the difference of the target and current location of the center (Figure 4.10). As elements with values might be shifted out from the matrix, the translation applied will not bring the center to the target location. The matrix is recursively translated until the error is smaller than a predefined threshold. As the sum of the elements can be smaller than one after some elements are shifted out from the matrix, the values are normalized at the end. This operation is defined here as $Tr(\mathbf{M}, \mathbf{c^d})$, where $\mathbf{c^d}$ is the target center location and $\mathbf{M}$ is the input matrix.

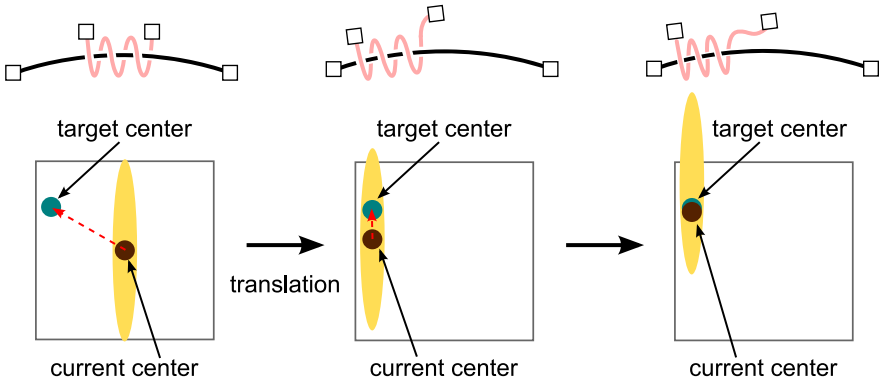


Figure 4.10: The elements of the writhe matrix are translated according to the difference of the target and current center location. As elements with values might be shifted out from the matrix, the translation applied will not bring the center to the target location. The process is repeated until the current center comes close enough to the target location.

4.3.1.3 Calculating the desired writhe matrix

Up to here, the writhe value has been kept to 1. Finally, we multiply w to the whole matrix so that it reaches the desired writhe value. Let us define this operation by $S(\mathbf{M}, w^d)$. We sequentially apply $R()$, $Tr()$ and $S()$ to $\mathbf{I}$ to obtain the writhe matrix of the target Topology Coordinate. It is to be noted that when these operators are applied to the writhe matrix in this order, each operation will only update either the density, center or the writhe, respectively, without affecting the other attributes. The writhe matrix $\mathbf{T}$ computed by the following operation represents Topology Coordinate $(w^d, d^d, \mathbf{c^d})$:

$$\mathbf{T^d} = S(Tr(R(\mathbf{I}, d^d - \frac{\pi}{4}), \mathbf{c^d}), w^d). \tag{4.3}$$

4.3.2 Mapping Topology Coordinates to Generalized Coordinates

Here we explain how to update the generalized coordinates according to the updates in of the Topology Coordinates. Assuming the current generalized coordinates of the two chains are $(\mathbf{q_1}, \mathbf{q_2})$, and the Topology Coordinates of the two chains are to be updated from $\mathbf{G} = (w, d, \mathbf{c})$ to $\mathbf{G} + \Delta\mathbf{G} = (w + \Delta w, d + \Delta d, \mathbf{c} + \Delta\mathbf{c})$, the task here is to compute the updates of the generalized coordinates $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$.

We first compute $\mathbf{T^d}$, the desired writhe matrix at $\mathbf{G} + \Delta\mathbf{G}$, using the method explained in Section 4.3.1, and then update the kinematics of the chains so that the re-

sulting writhe matrix becomes similar to the desired writhe matrix. Let us assume the writhe matrix of the current configuration is $\mathbf{T}$, and its update after changing the configuration is $\Delta\mathbf{T}$.

$$\mathbf{T} + \Delta\mathbf{T} - \mathbf{T^d} = \mathbf{0} \tag{4.4}$$

Here we assume the updates are small enough so that the relationships of $\Delta\mathbf{T}$, and $(\Delta\mathbf{q_1}, \Delta\mathbf{q_2})$ are linear:

$$\Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} + \frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\vec{\mathbf{q}}_\mathbf{2}. \tag{4.5}$$

When we control the two chains, we need to make sure that the two chains do not penetrate through each other. When two line segments approach to each other, the writhe value increases. At the moment before they cross each other, the absolute value of the writhe approaches 0.5. Therefore, penetrations can be avoided by limiting the maximum writhe value between arbitrary segments:

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma \quad (1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{4.6}$$

where $\Delta T_{i,j}$ is the (i,j)th element of $\Delta\mathbf{T}$, σ is a threshold, that is set to 0.2 in our experiments to prevent the segments from approaching too close to each other.

We can also add any kinematical constraints which can be linearized with respect to the generalized coordinates when the movement is small, such as the movements of any parts of the body in Cartesian coordinates or the center of mass. Let us summarize such constraints as follows:

$$\mathbf{r} = \mathbf{J_1}\Delta\mathbf{q_1} + \mathbf{J_2}\Delta\mathbf{q_2}, \tag{4.7}$$

where $\mathbf{J_1}, \mathbf{J_2}$ are the Jacobians of this constraint, and $\mathbf{r}$ is the linearized output of this constraint.

The update of the configurations of the chains can be computed by minimizing an objective function that represents the norm of the movements subject to constraints (4.5)-(4.7):

$$\min_{\Delta\mathbf{q_1}, \Delta\mathbf{q_2}, \delta} \|\Delta\mathbf{q_1}\|^2 + \|\Delta\mathbf{q_2}\|^2 + \|\delta\|^2 \text{ s.t.} \begin{cases} \Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} + \frac{\partial\mathbf{T}}{\partial\mathbf{q_2}}\Delta\mathbf{q_2} \\ |T_{i,j} + \Delta T_{i,j}| \leq \sigma \\ (1 \leq i \leq n_1, 1 \leq j \leq n_2) \\ \mathbf{T} + \Delta\mathbf{T} - \mathbf{T^d} + \delta = \mathbf{0} \\ \mathbf{r} = \mathbf{J_1}\Delta\mathbf{q_1} + \mathbf{J_2}\Delta\mathbf{q_2} \end{cases} \tag{4.8}$$

where δ is a vector of parameters introduced to convert Eq. 4.4 into soft constraints. The updated generalized coordinates $(\mathbf{q_1}+\Delta\mathbf{q_1}, \mathbf{q_2}+\Delta\mathbf{q_2})$ correspond to the target Topology Coordinate $(w+\Delta w, d+\Delta d, \mathbf{c}+\Delta\mathbf{c})$. By repetitively solving Eq. 4.8, we can update the configurations by specifying the trajectories of the Topology Coordinates.

In the proposed method, there is an unique mapping from the Topology coordinates to the desired writhe matrix. However, there is no reverse mapping from a writhe matrix back to the Topology Coordinates. Instead, we use the desired writhe matrix as a reference to control the actual writhe matrix of the links in order to control how they are tangled together.

4.3.3 Additional Manipulations

Here we first introduce a technique to pass a serial chain through a loop (Figure 4.11 (a)), which is useful for simulating movements such as moving the arms through sleeves or passing a hand through a hole. Next we introduce a technique to tangle a serial chain with a bundle of serial chains (Figure 4.11 (b)). Such a motion happens when holding the arms and torso of another character at the same time.

Passing a chain through loops: When controlling the serial chain to pass through a loop, we will only use the writhe of the Topology Coordinates as we do not care near which part of the loop that the serial chain is passing through. The writhe can be computed by Eq. 4.1. The writhe approaches one when the chain passes through the loop, as shown in Figure 4.11(a), (Recall the Gauss Integral computes the average number of crossings when viewing from all directions). Therefore, we control the serial chain by gradually increasing the writhe value between the loop and the chain and updating the kinematics of the serial chain accordingly:

$$\min_{\Delta\mathbf{q},\delta} \|\Delta\mathbf{q}\|^2+\delta^2 \;\text{ subject to }\; w+\frac{\partial w}{\partial \mathbf{q}}\Delta\mathbf{q}+\delta = w_d \tag{4.9}$$

where $\mathbf{q}, \Delta\mathbf{q}$ are the generalized coordinates of the serial chain and their updates, δ is a variable to convert the constraint into a soft constraint, w is the current writhe value between the chain and the loop and w_d is the target writhe value to reach in this step. When passing the chain through multiple loops, we switch the target loop from one to another. In our experiments, when the writhe of the chain with the current target loop has exceeded 0.95, we switched the target loop to the next one (Figure 4.11(a)).

Tangling a chain with a bundle of chains: Here we explain how to tangle a chain C with a bundle of chains $C_1,...,C_M$, while avoiding getting tangled with another set of chains $\bar{C}_1,...,\bar{C}_m$ (Figure 4.11(b)). The main idea is to specify the average of the

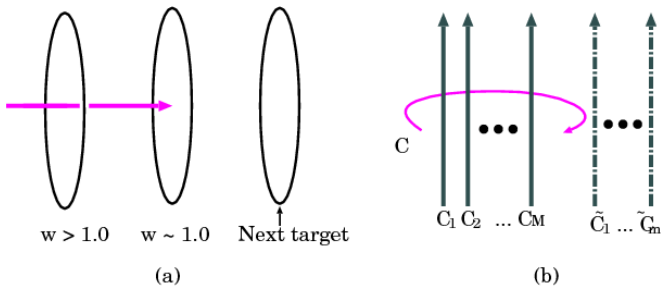


Figure 4.11: (a) Passing a chain through loops. The writhe is around 1 when the chain passes through the loop. The chain can be passed through multiple loops by sequentially switching the target loop. (b) Tangling a chain with a bundle of chains while avoiding to get tangled with others.

writhes between C and $C_1, ..., C_M$, while keeping their variance small. The following problem is repetitively solved until the chain twists around a bundle of chains:

$$\min_{\Delta \mathbf{q}} \|\Delta \mathbf{q}\|^2 + \sum_{k=1}^{M} \left(\frac{w+\Delta w}{M} - (w_k + \Delta w_k)\right)^2$$

$$\text{subject to} \begin{cases} \Delta w = \Delta w_1 + ... + \Delta w_M \\ \Delta w_k = \frac{\partial w_k}{\partial \mathbf{q}} \Delta \mathbf{q} (1 \le k \le M) \\ T_{i,k_j} + \Delta T_{i,k_j} \le \sigma (1 \le k_j \le n_k) \\ 0.5 \ge \bar{w}_l + \Delta \bar{w}_l \\ \Delta \bar{w}_l = \frac{\partial \bar{w}_l}{\partial \mathbf{q}} \Delta \mathbf{q} (1 \le l \le m) \\ T_{i,l_j} + \Delta T_{i,l_j} \le \sigma (1 \le k_l \le n_l) \end{cases} \tag{4.10}$$

where $\Delta \mathbf{q}$ is the increment of C's generalized coordinates, n_k, n_l are the number of segments composing C_k and $\bar{C}_l$, w_k $(1 \le k \le M)$ and $w_l (1 \le l \le m)$ are writhes between C and $(C_k, \bar{C}_l)$, respectively. The third and sixth constraints keep the line segments from penetrating through each other. The fourth constraint prevents C from getting tangled with $\bar{C}_j$ by keeping their writhes below 0.5.

Here we again only specified the writhe of the Topology Coordinate as the chain may not have enough degrees of freedom for the user to specify the center and density, as there are many other chains to get tangled with. However, it is possible to combine this problem with that of Eq. 4.8 by specifying the Topology Coordinates of the controlled chain and one of the serial chains in the bundle.

4.4 Experiments

In this section, we show some example motions produced by our system. We first explain about our user interface to produce keyframe postures for character animation. Then, we show the resultant animations produced by interpolating such keyframe postures.

4.4.1 Controlling Characters

We provide an interface (Figure 4.22) for the user to interactively edit the postures of one or two characters. The user can specify the topological constraints as well as ordinary kinematic constraints, such as the trajectories of the end effectors. Once the keyframe postures are generated, their Topology Coordinates are available, and we can interpolate the postures in Topology Space.

Because the techniques explained in Section 4.3 are only for open serial chains or loops, we need to specify the paths between the character's joints / end effectors which we want to tangle. For instance, suppose we want to generate a posture of a wrestling attack called Full Nelson Hold (Figure 4.12, right), in which the attacker needs to first pass the arms under those of the other character from behind, and then let the attacker press the other's neck. In this case, the paths "left shoulder - left hand" and "right shoulder - right hand" must be twisted with the paths "head-top - left hand" and "head-top - right hand" of the other character, respectively. Once the paths to be tangled are

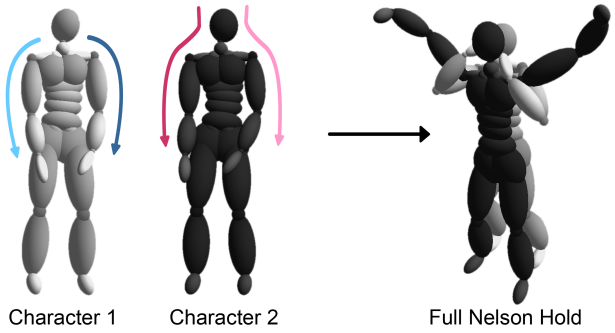


Figure 4.12: The user specifies the area of the body to be tangled. The two paths (left shoulder - left hand), (right shoulder - right hand) of Character 1 are to be twisted with (head-tip - left hand), (head-tip - right hand) of Character 2. The posture on the right is the expected final posture.

specified, the user can use the scrollbars to adjust the writhes, density and center of the

paths. The values set by the scrollbars are used to compute the desired writhe matrix $\mathbf{T}^{\mathbf{d}}$ in Eq. 4.8.

If there are two characters, we constrain the posture of one character by default, as we found this is easier for the user to generate the desired postures. The user can alternately switch the active character until the postures are satisfactory. The user can also adjust the postures using inverse kinematics. In this case, the posture of the other character is updated so that the Topology Coordinates are kept the same.

The postures created can be used as a keyframe for motion synthesis. We interpolate the keyframes in Topology Space, and therefore, we can easily twist the segments around each other without using complex path-planning or collision avoidance schemes. The interpolated motions can be further adjusted by the user using inverse kinematics: the user can drag a segment of one of the bodies by the mouse. The postures of the two bodies will be updated so that the kinematic constraints and topological constraints are both satisfied.

4.4.2 Motion Synthesis

Here, experimental results of controlling characters by the Topology Coordinates are shown. First, motions of a single / multiple human characters that require close contacts between segments were created. Next, examples of a human character interacting with rigid / deformable objects were created. Finally, animations of an octopus interacting with swimming fishes or a human character were created.

Human character animation : First, a motion of a single character was generated by interpolating keyframe postures shown in Figure 4.13(a). In these postures, the body of the character is self-tangled. Such postures cannot be interpolated by the generalized coordinates without self-collisions. We can easily produce natural-looking behaviors by interpolating them in Topology Space.

Next, wrestling movements of two human characters were simulated by interpolating five keyframe postures in Figure 4.13(b). For the motion of the red character in the front, we imported motion capture data of a single character moving around. The Topology Coordinates of the postures were linearly interpolated. The attacking character behind had to dynamically adjust its movements so that the Topology Coordinates between the limbs become the same as the target values. This example shows that our method can be combinedly used with captured motion data.

Finally, a piggyback motion that requires multiple tangles by the limbs was created

as shown in Figure 4.13(c). Four keyframe postures were used to create this motion. Although this motion requires a significant amount of collision avoidance, we can produce the motion in a very short time. The computational cost for creating this motion is compared with that when using a global path-planning method based on RRT is presented in Section 4.4.4.2.

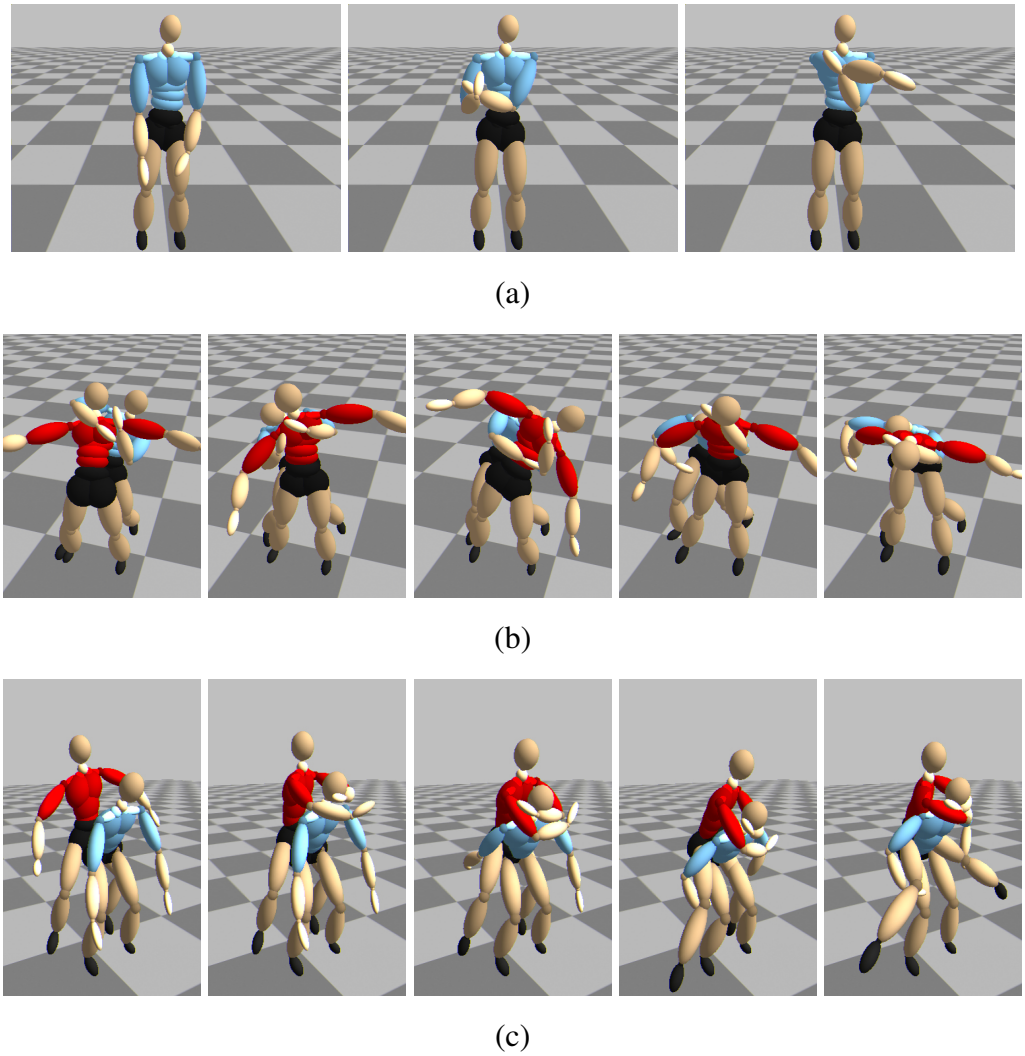


Figure 4.13: (a) Three keyframe postures to generate a stretching motion. (b) The wrestling motion in which the red character re-holds the blue character in various ways. (c) A piggyback motion created from four keyframes.

Interaction with rigid/deformable objects: First, we created an animation of a character re-holding a chair in various ways. The topological structure of the chair is shown in Figure 4.14 (left). The five keyframes shown in Figure 4.16 are used to produce the animation. The keyframes are created by changing the Topology Coordinates of the character's arm with the pipes of the chair. If these postures are interpolated by gener-

alized coordinates, the body can easily penetrate through the pipes of the chair. When we interpolate the postures in Topology Space, the arms can avoid the pipes and hold the chair. Next, an example of a character holding a deformable object composed by

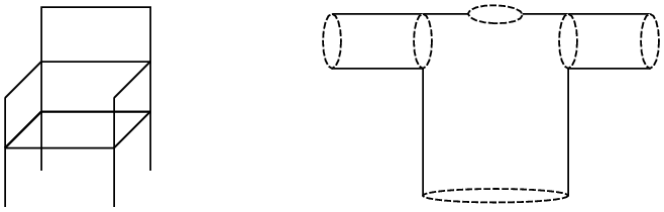


Figure 4.14: The topology of the chair model composed of eighteen pipes (left) and the shirt model with six rings (right) used for creating the animations.

a bundle of line segments is generated (Figure 4.19(left)). The writhe values between the path formed by the two arms and the bundle of lines of the object were increased to create an animation of holding the object. The method explained in Section 4.3.3 was used to simulate such a motion. The character can stably hug the object although the shape of the object is dynamically deformed. The character adjusts its posture so that its topological relationship with the object does not change. This example shows our method is extensible to control the topological relationships of 3D shapes.

Finally, an example of a human character wearing a shirt is made. Six virtual rings were prepared inside and at the fringe of the shirt (Figure 4.14 (right)). The character was guided to pass its arms and necks through the rings by the topological constraints. The screenshots of the animation are shown in Figure 4.15. This example shows that our method is also useful for creating animations of characters interacting not only with each other, but also with deformable models. It also shows we can handle topological relationships between the serial bodies and rings.

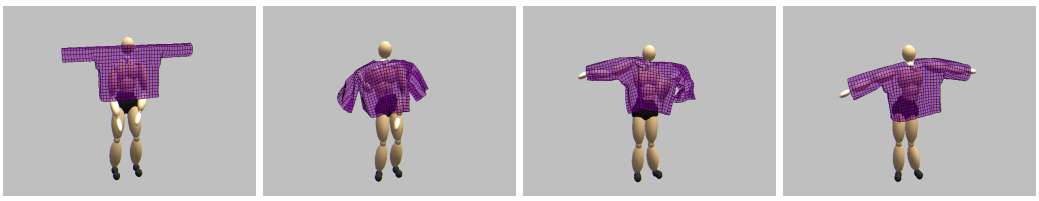


Figure 4.15: An animation of a human character wearing a shirt.

Octopus Motions: We have prepared an octopus model whose legs are modeled by serial chains of rigid segments. Each leg is composed of twelve segments. Firstly, we show an example in which the octopus catches a number of fishes using different limbs simultaneously (Figure 4.17). The difficulty of generating such motions is that

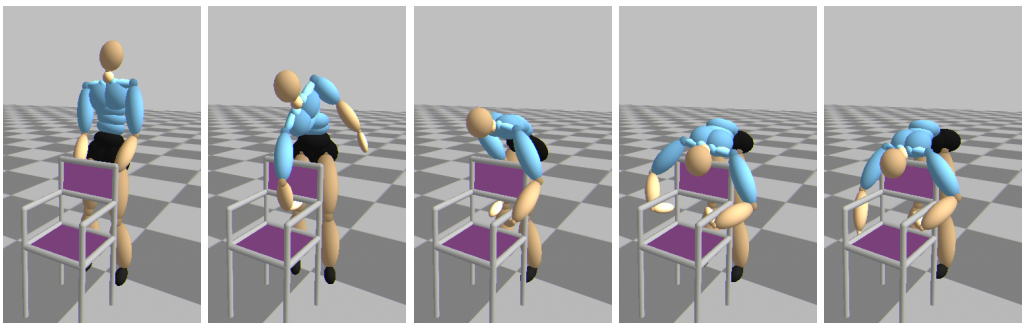
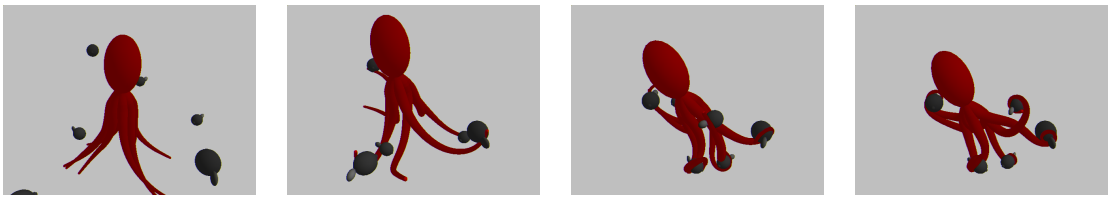


Figure 4.16: The five keyframes to produce the animation of re-holding the chair.

if we do not take into account the tangle constraints, the limbs can get tangled with each other. In our system, this is avoided by adding extra constraints to prevent the limbs to generate tangles whose writhe is larger than 0.5. As a result, the limbs will move away from each other when their distance gets closer. We have also generated an example of tangling the octopus with a human character (Figure 4.18). The motions of the octopus were created from only two keyframe postures, one each for the initial and final posture.



(a) (b) (c) (d)

Figure 4.17: An animation of an octopus catches a number of fishes created by two keyframes (the initial (a) and final (d) posture).

(a) (b) (c) (d)

Figure 4.18: An animation of an octopus tangles with a human character created by two keyframes (the initial (a) and final (d) posture).

In summary, although these animations can be produced by existing techniques, they will require careful tuning to avoid artifacts. For example, if they are made by tra-

ditional keyframe animation methods, a great number of keyframes need to be inserted to guide the character correctly and avoid penetrations of segments.

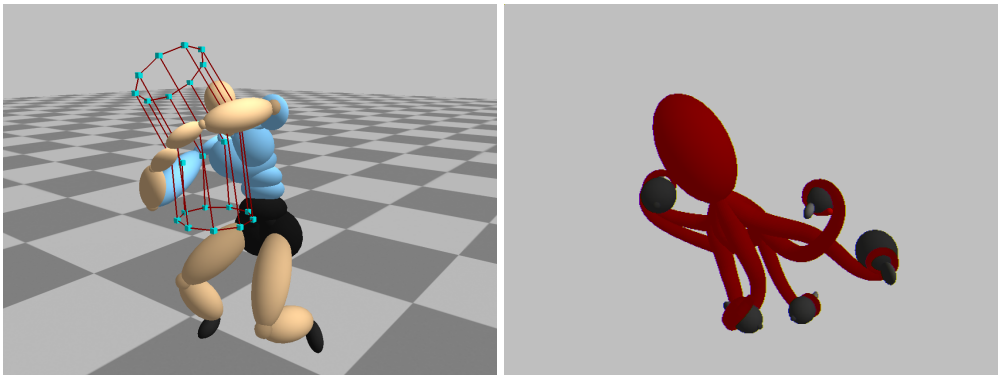


Figure 4.19: A character hugging an object composed of a bundle of line segments (left), and an octopus catching multiple fish simultaneously using its limbs (right)

4.4.3 Computational Costs

In our demos, we used a human character model of 42 DOF, and an octopus model of 276 DOF. All DOFs are used when controlling the characters. For the human character model, when one character is controlled, we can obtain an interactive rate of 40 frames per second when using a Pentium IV 3.2 GHz PC. When two characters are simultaneously controlled, we can still obtain a frame rate of 30 frames per second. We use ILOG CPLEX 9.1 [ILOG Inc, 2005] as our quadratic programming solver.

The bottleneck of the computation is at that of the writhe matrix. When we want to tangle a body of n_1 segments with that of n_2 segments while avoiding it getting tangled with another body of n_3 segments, the cost for computing the writhe matrices becomes $O(n_1 \times n_2 + n_1 \times n_3)$. For tangle avoidance, we can omit computing the matrix elements for those which are far away from each other. Therefore, the complexity becomes $O(n_1 \times n_2)$. The analytical solver of the GLI enables fast computation of the writhe matrix. As the writhe matrix is sparse, once it is computed, the quadratic programming solver can efficiently compute the generalized coordinates. Therefore, we can achieve real-time performance even when the DOFs of the bodies are large.

4.4.4 Comparison with global path planning algorithms

We have conducted an experiment to compare the performance of synthesizing tangling motions using Topology Coordinates with global path planning methods in joint

angle space. In [Ho and Komura, 2007a], we proposed to use GLI as one of the criteria when planning tangling motions using Rapidly-Exploring Random Trees (RRT). The details of the algorithm are explained in the next section (Section 4.4.4.1).

4.4.4.1 Motion Planning by the RRT

Now we explain the algorithm to plan the given configuration from postures at which the two avatars are separate. There are two assumptions in this method: 1) the planning starts from postures such that none of the segments are tangled; 2) the characters are standing close enough to each other so that they can start the segments to get tangled. Let us assume the information of all the tangles in the target configuration q_{goal}, which is the final posture of the characters in the animation, is saved in a list. We pick a tangle from the list and move the characters so that the GLI of the path composing the tangle becomes close enough to that in q_{goal}. This is repeated for all the tangles.

The movements of the characters to compose each tangle is computed using the RRT method. This is a motion planning problem to find out a collision-free path starting from the initial position q_{init} to the target posture q_{goal}. Bertram et al. [Bertram et al., 2006] modified the RRT searching algorithm for reaching tasks on robot arms by guiding the RRT expansion towards the goal region instead of defining the goal configurations explicitly. We use this approach to make the character entangle its body to the other character.

At each step of the RRT search, the 3D motion of end effector E, which is mainly involved in the tangle, is computed. The posture of the character is then computed by inverse kinematics (IK). Only collision-free body configurations are valid for the RRT expansion.

The pseudo-code of the modified RRT planning algorithm BuildRRT is shown in Algorithm 2 and 3. The algorithm is almost the same as those described in Yamane et al. [Yamane et al., 2004] and Bertram et al. [Bertram et al., 2006].

In BuildRRT(), starting from the initial posture q_{init}, the RRT τ is expanded until a posture that satisfies the preferred topological relationship is found. At each iteration, the function SelectTarget() returns a random 3D location. However, it returns the target configuration q_{goal} at a probability of 0.05 [Yamane et al., 2004]. This output point V_{rand} of SelectTarget() is passed to subfunction ExtendHeuristics() where the RRT τ is further expanded.

In ExtendHeuristics(), given the random point V_{rand} in 3D space, we first find the configuration q_{near} at which the position of the end effector is closest to V_{rand}.

Starting from q_{near}, the new posture q_{new} is then computed in NewConfigByIK(). The system uses a general IK engine to compute the new posture from q_{near} by translating the end effector towards V_{rand} for a predefined step size.

Algorithm 2 BuildRRT($q_{init}, q_{goal}, Tangle(p_a, p_b)$)

$E \leftarrow$ End Effector of p_a
τ.init(q_{init});
for k = 1 to K **do**
  $V_{rand} \leftarrow$ SelectTarget($V(E, q_{goal})$, 0.05);
  ExtendHeuristics(τ, $V_{rand}, Tangle(p_a, p_b)$;
end for
return τ;

In NewConfigByIK(), collision detection is performed on the body segments of the characters. As we are planning the movement of only the end effector, there will be more than one IK solutions in most cases. However, only collision-free body configurations are valid for the RRT expansion. Once a collision is detected, the system will try to edit the positions of the body segments according to the penetration depth while satisfying the constraints of translating the end effector. If there no collision-free configuration is found, q_{new} will be discarded.

After IK, the GLI value of the paths between the bodies are computed to check whether the tangle's GLI value is getting closer to that of the target configuration. We have to also make sure that tangles which do not exist in the target configuration are not generated when expanding the RRT. In order to avoid configurations with extra tangles, we monitor the GLI values between the path P_a and all the P_bs of the other body: if any GLI goes above 0.5 for pairs which are not tangled in the target configuration, this configuration is discarded. If these conditions are satisfied, the new configuration q_{new} is added into the branch of τ. By comparing the GLI values, the RRT expansion will be biased to create a body configuration which is similar to q_{goal}.

4.4.4.2 Results

For the scene of one character piggybacking another (Figure 4.13 (c)), it takes more than 1000 seconds when the global path planning method presented in Section 4.4.4.1 is applied. On the other hand, by using Topology Coordinates, we can obtain the results in less than 2 seconds.

Algorithm 3 ExtendHeuristics($\tau, V_{rand}, Tangle(p_a, p_b)$))

$q_{near} \leftarrow$ ranking.front();
if NewConfigByIK(V_{rand}, q_{near}, q_{new})) returns a valid q_{new} **then**
  for every path p'_b in Body B which is not included in *TangleList* **do**
    if $|GLI_{q_{new}}(p_a, p'_b)| > 0.5$ **then**
      Goto FAILURE;
    end if
  end for
  if $|GLI_{q_{new}}(p_a, p_b) - GLI_{q_{goal}}(p_a, p_b)| < |GLI_{q_{near}}(p_a, p_b) - GLI_{q_{goal}}(p_a, p_b)|$ **then**
    return q_{new}
  end if
end if

FAILURE:

IncreaseFailureCount(q_{near}, 1);
if FailureCount(q_{near}) $> f$ **then**
  ranking.remove(q_{near});
  $q_{parent} \leftarrow$ PARENT(q_{near});
  IncreaseFailureCount(q_{parent}, 1);
end if
return q_{near};

4.4.4.3 Discussion

Now let us discuss about the advantage of using Topology Coordinates for motion planning. Topology Coordinates provide a high-level abstraction of the relationship between the body segments and they can be easily converted into the generalized coordinates by solving a quadratic programming problem in polynomial time. Continuous trajectories at the level of Topology Coordinates will always result in continuous movements in the motion space. On the contrary, existing representations based on joint angles / positions greatly suffer from the expensive cost of examining the validity of every transition which can be easily violated by collisions and penetrations of the body parts. This is due to the fact that the Topology Coordinates is a representation based on the relationships, while existing representations requires an additional step to evaluate the relationships after the actual posture of the characters are computed.

Let us examine the complexity of planning a basic movement of winding one arm around another. If we are using Topology Coordinates, we simply need to update their values by linear interpolation, which results in the complexity of $O(N)$ for solving the quadratic programming problem in Eq. 4.8, where N is the degrees of freedom of the body. On the contrary, for directly exploring for the target posture by RRT at the level of joint angles, the cost will be $O(exp(N))$. In addition to that, the cost of each iteration, will also be expensive as the test of collisions and pass throughs is further required whenever a new random posture is produced.

Now, let us examine the complexity for more complex target postures that require the winding of multiple limbs, such as those in wrestling movements. In this case, the cost of planning for Topology Coordinates can increase to $O(n! \times N)$ where n is the number of windings between the limbs. This is assuming that every wind will be produced one after another, which is usually the case for humans tangling their bodies with others. For methods based on joint angle + RRT, the cost will remain the same as $O(exp(N))$ as it simply randomly expands all the DOF simultaneously. Although the theoretical cost for Topology Coordinates may increase due to the permutation of the tangle order, in practice such effect is negligible for character animation due to the low number of tangles that can be produced between the bodies. In addition to that, extra constraints (for example, In case of the piggy-back, the arms of the person to be carried need to be tangled with the neck of the carrier first to increase the stability) will further bring down the number of the possible sequences.

To summarize, motion planning at the level of Topology Coordinates can greatly reduce the cost of motion planning thanks to the abstractness and the relationship-based representation.

4.5 Real-time Character Control for Wrestling Games

Now we introduce a technique to make use of the Topology Coordinates to simulate complex interactions such as those in wrestling games in real-time. A finite state machine of attacks and defenses based on Topology Coordinates is precomputed and used to present the game players the next possible moves for attacking / escaping from the opponent. The player who has an opportunity to attack is shown the possible actions on the list of icons. The characters can also be controlled kinematically by using inverse kinematics.

We first prepare a finite state machine of two characters wrestling based on Topology Coordinates. This is produced by the animator by first keyframing a number of representative postures of wrestling attacks by Topology Coordinates, and then connecting postures by transitions that correspond to movements of entangling / desolving tangles.

During runtime, the Topology Coordinates of the two virtual wrestlers are computed based on their postures, and the corresponding node in the finite state machine is found. The players are then shown the list of possible attacks that can be launched from the current posture. The user also has a choice to control the characters directly by inverse kinematics to get away from attacks and holds by the opponent player.

Let us first give an overview of the computation of the characters' motions. The flow of the system is illustrated in Figure 4.20. The motion of each character is computed sequentially by two different quadratic programming problems. The attacker's movement is guided by the Topology Coordinates - once the attack is specified, the attacker tries to tangle its body with the defender's body according to the target configuration. The attack can be switched in the middle if the player thinks a different attack is more effective under the current configuration. The defender needs to escape from the attacks by controlling the body segments involved in the tangling process. The player uses a pointing device to move a body segment and the posture of the defender is computed by inverse kinematics based on quadratic programming.

Now we explain about the method to control the attacker. The attacker's motion is determined based on the defender's current posture and the target Topology Coor-

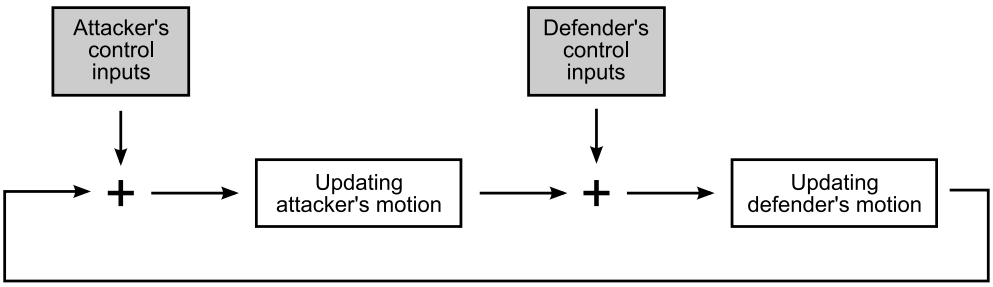


Figure 4.20: Overview of the character motion synthesis loop. The motions of the attacker and defender are computed sequentially by two different quadratic programming problems.

dinates in the next time step. Let us assume the configurations of the attacker and defender are represented by $\mathbf{q_1}$ and $\mathbf{q_2}$, respectively. Solving Eq. 4.8 is not a good idea to compute the motion of the attacker, as the updates of the attacker's movements takes into account the movements of the defender in the next time step. This makes the defender difficult to escape from the attacker. Therefore, we solve the following problem for computing the motion of the attacker:

$$\min_{\Delta\mathbf{q_1},\delta} \|\Delta\mathbf{q_1}\|^2 + \|\delta\|^2 \text{ s.t.} \tag{4.11}$$

$$\Delta\mathbf{T} = \frac{\partial\mathbf{T}}{\partial\mathbf{q_1}}\Delta\mathbf{q_1} \tag{4.12}$$

$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma(1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{4.13}$$

$$\mathbf{T} + \Delta\mathbf{T} - \mathbf{T^d} + \delta = \mathbf{0} \tag{4.14}$$

$$\mathbf{r_1} = \mathbf{J_1}\Delta\mathbf{q_1} \tag{4.15}$$

where $\mathbf{r_1}$ represents the kinematic parameters of the attacker. This technique adds an effect of physiological delay for launching a response motion with respect to the defender's movements, which increases the realism of the interaction. It also adds an essence of a game play to the interaction between the two virtual wrestlers as the attacker may not be able to achieve the target Topology Coordinates if the defender is controlled well.

Next, the defender's movement is computed by solving the following inverse kine-

matics problem:

$$\min_{\Delta \mathbf{q_2}} \|\Delta \mathbf{q_2}\|^2 \text{ s.t.} \tag{4.16}$$

$$\Delta \mathbf{T} = \frac{\partial \mathbf{T}}{\partial \mathbf{q_2}} \Delta \mathbf{q_2} \tag{4.17}$$

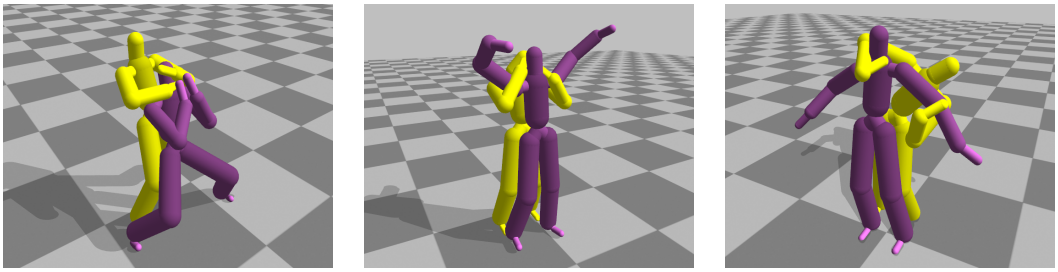
$$|T_{i,j} + \Delta T_{i,j}| \leq \sigma (1 \leq i \leq n_1, 1 \leq j \leq n_2) \tag{4.18}$$

$$\mathbf{r_2} = \mathbf{J_2} \Delta \mathbf{q_2}. \tag{4.19}$$

where $\mathbf{r_2}$ represents the kinematic parameters of the defender, based on the input from the pointing device and any other kinematic constraints. The GLI between every segment pairs are still considered to avoid penetrations.

The attacker's motion is computed first by solving Eq. 4.11 and the attacker's posture is updated. Then, the defender's motion is computed by solving Eq. 4.16. Once both character's motion is updated, the time counter is increased.

The key point of controlling the defender is to move the body such that it can efficiently escape from the attacks. Such movements are those which can reduce the writhe value by little motion. For example, when the attacker starts to shift to a configuration of a rear-choke hold (Figure 4.21 (a)) , which is an attack to squeeze the neck from behind, the most efficient way to escape is to knee down at the last moment, which requires little movement. On the other hand, if the attacking player can predict such escaping moves of the defender and switch to another move that can make use of such movements, the player can efficiently tangle the attacker's body to the defender.



(a) (b) (c)

Figure 4.21: Various wrestling interactions created by the proposed method.

4.5.1 Finite State Machine for Wrestling

In this subsection, we explain how a Finite State Machine (FSM) of two characters wrestling are produced from the keyframe postures designed using the Topology Coordinates.

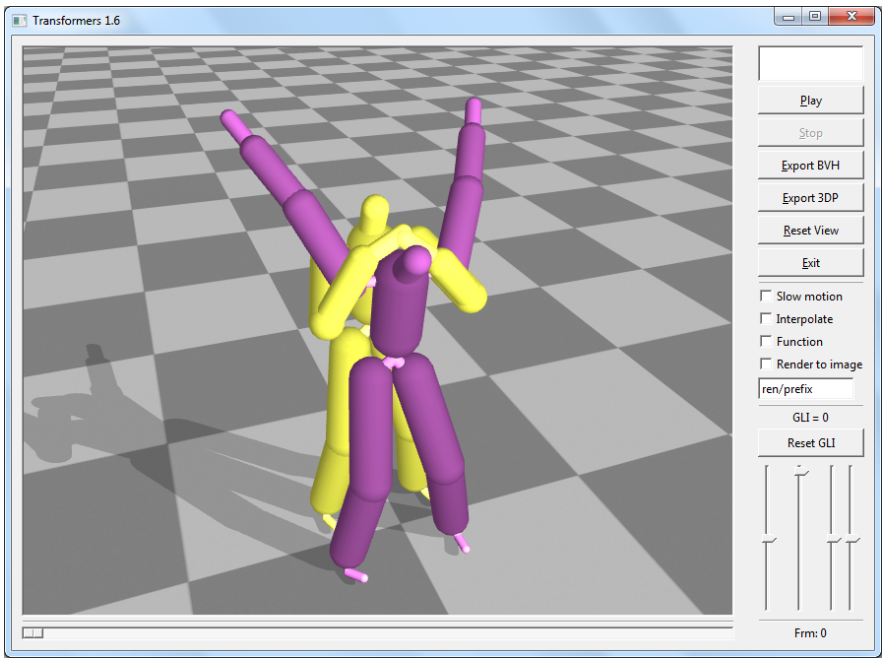


Figure 4.22: An interface to edit the postures of the wrestling characters by the Topology Coordinates. The animator specifies the limbs to be tangled and adjust their Topology Coordinates by the scroll bar at the right.

The animator first prepares postures of two characters wrestling by updating the Topology Coordinates of the body segments. Scroll-bars that let the user adjust the writhe, center and density values are provided to generate different configurations. A view of the interface for editing the postures by the Topology Coordinates is shown in Figure 4.22.

Once the postures are designed, we start to produce the FSM. First, the designed postures are added as new states of the FSM. The topological status of the designed postures are evaluated based on the concept of rational tangles presented in Chapter 3. The rational tangles between all the routes that connect the end effectors are computed. Next, we find all the shortest paths from the designed postures to the untangled states in which the two characters simply stand next to each other. The limbs are untangled one by one, and each of such states are added into the state machine as well. Let us call these states *intermediate states*. Many intermediate states might be shared between different attacks.

Finally, the system creates the FSM by connecting the nodes (postures) with similar topological states. As discussed in Chapter 3, we connect the postures/states if the absolute differences of the writhe between every pair of routes are less than a threshold of 0.5. An example finite state machine of one character attacking another from behind

is shown in Figure 4.23.

The FSM mode starts only when the characters are close enough and one of the player launches an attacking action. When the characters are separate from each other in the beginning, they are not in the state of the FSM. Both characters freely move around according to the user input until either of them reach a state where the user can launch a new attack. As shown in Figure 4.24, the possible choices are shown on the screen and the user selects one of them by the mouse.

There are also states that the defender can start an attack - if the player controlling the defender reacts faster than the player controlling the attacker to launch such an attack, the status of the fight can be switched. The user who has selected an action earlier becomes the attacker.

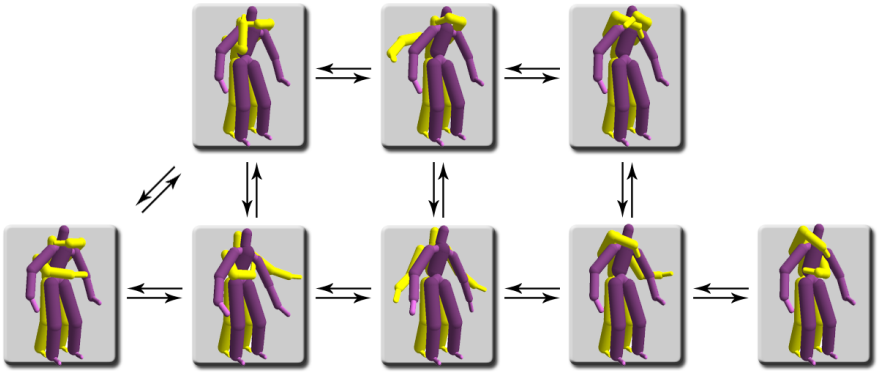


Figure 4.23: A finite state machine of two people wrestling when one character at the back of the other.

4.5.2 Experimental results

We have simulated a number of interactions between two virtual wrestlers both controlled by game players. Those include rear-choke hold (Figure 4.21 (a)), Full Nelson hold (Figure 4.21 (b)) and a number of different squeezing motions from the back. The virtual wrestlers start from separate postures and the attacker approaches to the defender to tangle its body with it.

The movement of the attacker is controlled by specifying the Topology Coordinates. In the first animation, the attacker performed an attack by tangling i) right arm with the neck of the defender and ii) left arm with the left arm of the defender. Without controlling the defender, this attack can be performed easily by changing the Topology Coordinates of the attacker as shown in Figure 4.25.

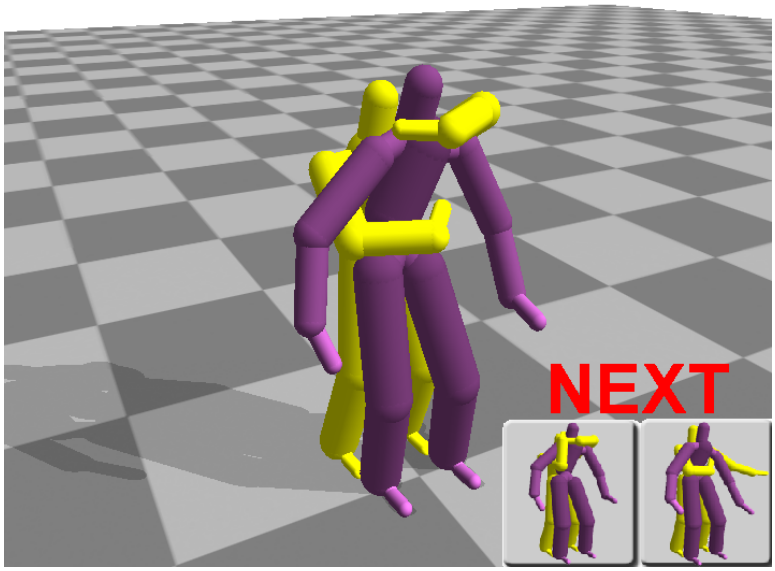


Figure 4.24: Once an intermediate state of the FSM is reached, the possible transitions are shown to the user.

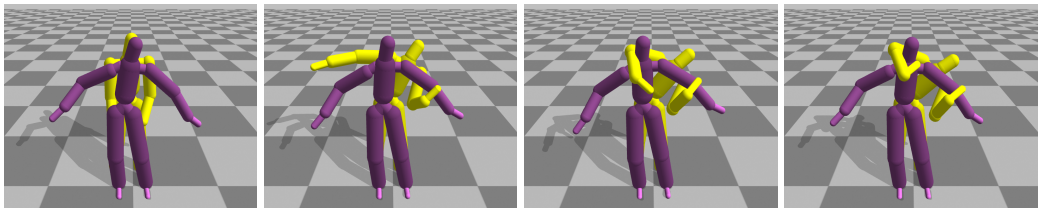


Figure 4.25: Without controlling the defender (in purple), the attacker (in yellow) can tangle with the defender easily by changing the Topology Coordinates.

In the second example, the defender tries to escape from the attacks shown in Figure 4.25. In our implementation, the player can specify kinematic constraints on the defender by dragging the body segments using a pointing device. In Figure 4.26, the player dragged the left hand (colored in red) of the defender. However, the defender fails to escape from the attack as it was not controlled quick enough. More examples are shown in Figure 4.27. In Figure 4.28, the defender escapes from the attack successfully by moving the torso quickly and vigorously.

The attacker can also switch to another wrestling move in the middle of the attack. In the third example, the defender escapes from the attack by blocking the right arm of the attacker in the early stage of the interaction. Then the attacker switches to another wrestling move by tangling its right arm with the right arm of the defender and its left arm with the torso and neck of the defender. Finally the attacker successfully locks

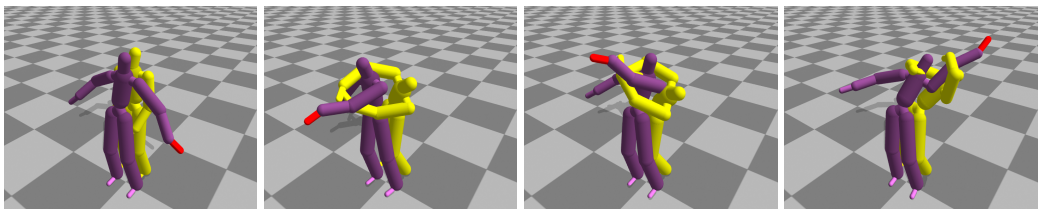


Figure 4.26: The defender (in purple) cannot escape from the attack if the player does not control it quick enough.

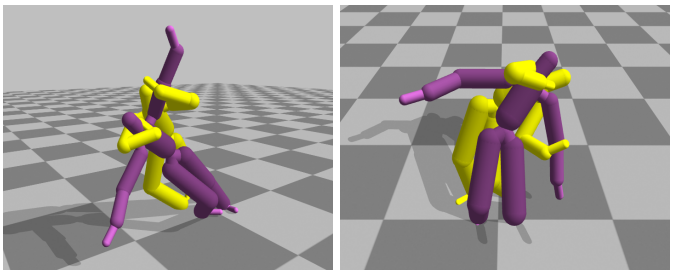


Figure 4.27: The defender (in purple) cannot escape from the attack if the player does not control it quick enough.

with the defender. The screenshots of this interaction are shown in Figure 4.29.

4.6 Summary and Discussions

In this chapter, we have proposed a new method to control characters by using topological constraints. We specify how the segments of the characters should twist around each other over time. Various basic motions such as hugging, wrestling attacks, holding a bulky object, which were difficult to be handled before can be generated. The ability to create complex motions of a character which has a large DOF such as an octopus tangling its arms with a human character while avoiding the penetrations from only a few number of keyframes is a great advantage of our method. Existing global path-planning techniques based on RRT will suffer when DOF of the character is large.

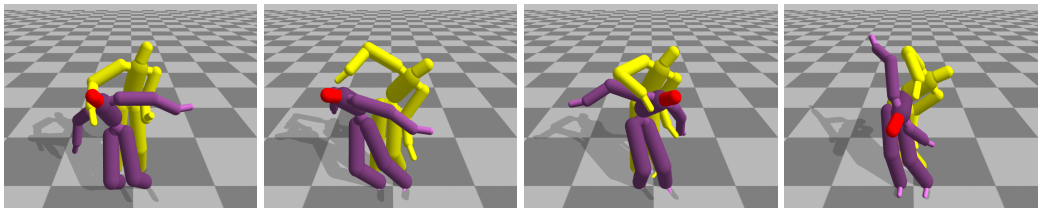


Figure 4.28: The defender (in purple) escapes from the attack.

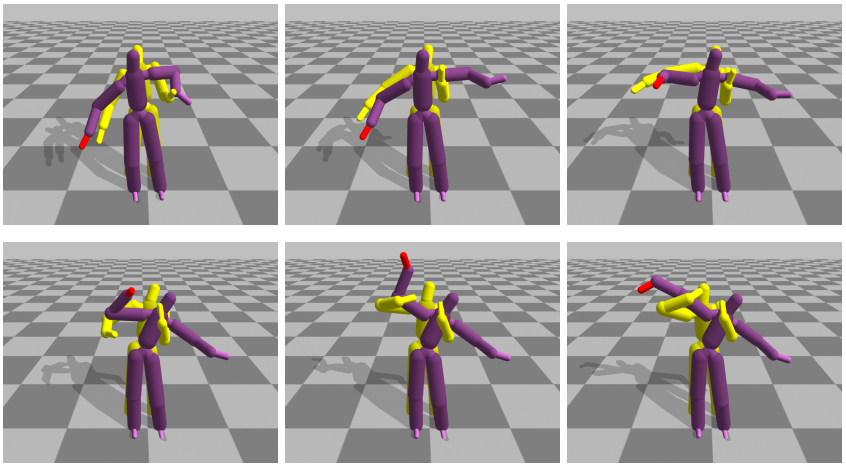


Figure 4.29: The attacker (in yellow) switched the attack in the middle in order to lock the defender (in purple).

There are some shortcomings with our method: First of all, as our method is based on local optimization, in some cases the solver can get caught in local minima. This happens mostly when there are too many topological constraints to be solved simultaneously. In fact, some topological constraints cannot be physically satisfied at the same time. In such a case, the user needs to decrease the number of topological constraints by limiting the number of chains to get tangled with or restrained from being tangled with. This can be done interactively by the user.

Secondly, in this chapter we assume all the models are only composed of line segments. Many objects have area and volume, and sometimes it is difficult to approximate the topological relationships by those objects by line segments. We also did not handle the collisions between the rigid objects. The penetration of line segments can be examined by tracing the writhe value. However, another collision detection scheme is required for rigid bodies. Although we can insert a collision detection stage after the postural updates and add repulsive forces to the segments so that they do not penetrate each other, sometimes the shape of the rigid bodies can inhibit the bodies from getting tangled / untangled. One solution is to model each rigid segment by a mesh of line segments. We can use a multi-resolution approach and represent the segments by line segments at low resolution and by a mesh structure in high resolution. When precise collision avoidance based on the detailed shape is required, we can calculate the motions using the high resolution model.

Thirdly, the user needs to specify the order to tangle the segments to those of the other. When there are many tangles between the characters, there are many ways to

arrive to the goal postures. The appropriate way to evaluate such routes can be application dependent, and usually the user prefers to specify it by him/herself. Therefore, in this research, we let the user specify the sequence by him/herself by providing the list of sequence. Planning the sequence of tangles and synthesizing a sequence of motions can be one direction for future research.

As a future work, we would like to apply Topology Coordinates to analyze and synthesize knotting and unknotting motions of robots. Previous work in knot tying / 1D deformable object manipulations in the robotics community analyze the configuration / state of the knotting / tangling based on 2D configurations of the strands [Wakamatsu et al., 2006, Matsuno et al., 2006]. One of the limitations for these methods is that a 2D plane has to be defined in order to correctly compute the topological status. As a result, the work in [Wakamatsu et al., 2006, Matsuno et al., 2006] perform knotting / tangling on a table. However, it is very difficult to define such as plane when handling objects in the 3D world. As our method can analyze the topology of the configuration of strands in 3D, we believe that it is more flexible for planning interactions between robots and 1D deformable objects in the real world.

Chapter 5

Spatial Relationship Preserving Character Motion Adaptation

While interactions with complex tangling of body segments such as those in wrestling and dancing can be efficiently generated by Topology Coordinates, we would like to generalize the concept of topology-based motion editing method and apply it to handle close interactions without any tangles.

In this chapter, we present a new method for editing and retargeting motions that involve close interactions between body parts of single or multiple articulated characters, such as dancing, wrestling, and sword fighting, or between characters and a restricted environment, such as getting into a car. In such motions, the implicit spatial relationships between body parts/objects are important for capturing the scene semantics. We introduce a simple structure called an *Interaction Mesh* to represent such spatial relationships. By minimizing the local deformation of the Interaction Meshes of animation frames, such relationships are preserved during motion editing while reducing the number of inappropriate interpenetrations. The Interaction Mesh representation is general and applicable to various kinds of close interactions. It also works well for interactions involving contacts and tangles as well as those without any contacts. The method is computationally efficient, allowing real-time character control. We demonstrate its effectiveness and versatility in synthesizing a wide variety of motions with close interactions.

Portions of this chapter have previously been published as [Ho et al., 2010].

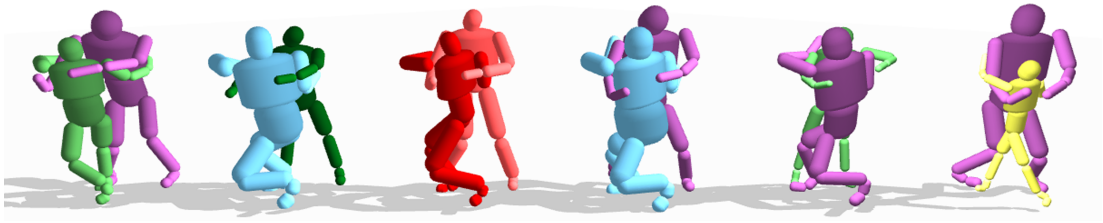


Figure 5.1: Our system can retarget motions of close interactions to characters of different morphologies. A judo interaction (red / orange pair) retargeted to characters of different sizes.

5.1 Introduction

Close interactions, not necessarily with any contacts, between different body parts of single or multiple characters or with the environment are common in computer animation and 3D computer games. Yoga, wrestling, dancing and moving through a constrained environment are some examples. In such motions, the spatial relationships between different body parts of characters are important in capturing the semantics of the scene. When an animator synthesizes or edits such movements, special care is needed to preserve these spatial relationships, for example, "arching back to avoid a punch", "hands extending around each other", "two bodies moving synchronously in close proximity" or "getting into a small car by bending down". However, traditionally, such spatial relationships exist only in the animator's mind and are not digitally embedded into the data. Although humans use spatial relationships to recognize semantics of interactions, their usage has not been considered much in character animation.

Existing scene representations have a fundamental limitation in handling such close interactions. Currently, a motion is typically described in terms of joint angles and kinematic constraints such as contacts. With this representation, automatically computing a valid motion requires randomized exploration and significant computation for collision detection. The animator also needs to shoulder the burden of specifying all the kinematic constraints in advance. From the animator's perspective, this is impractical and not conductive to manual editing. Competitive automatic solutions require an effective representation that allows the extraction of spatial relationships from existing motion data and synthesis of new animations that preserve these relationships. Such a representation will not only allow quantitative evaluation of the way different body parts are interacting, but also facilitate qualitative characterization of scene semantics.

In this research, we propose a simple representation which we call the *Interaction*

Mesh to represent the spatial relationships between nearby body parts. The Interaction Mesh is a volumetric mesh defined by the joints of the characters and the vertices of the objects/environment with which the characters are interacting. When editing or retargeting the movements, the motions are automatically adapted by deforming the Interaction Meshes at all frames with efficient Laplacian deformation techniques [Alexa, 2003, Zhou et al., 2005]. The high-level semantics of the interactions are maintained through preserving the local details of the Interaction Meshes.

The Interaction Mesh representation is general. It provides a unified treatment for interacting body parts of single or multiple characters as well as objects in the environment. As a result, it is applicable to many types of scenarios, such as when single character's actions involve close interactions between different body parts (dancing) or multi-character interactions (wrestling, fighting games). Additionally, the motions may either involve much tangling and contacts (e.g. judo, Figure 5.1) or little contact (e.g. Lambada dance). Additionally, it can be applied either per-frame or in the space-time domain according to the complexity of the problem and the available computing resources.

Motion adaptation with the Interaction Mesh is fully automatic. When the animator changes the size or morphology of the characters or edits parts of the motion, the system automatically deforms the Interaction Meshes at all the frames using a spacetime optimization and creates a new motion sequence that preserves the original context of the scene. No constraints need to be specified by the animator since they are all encoded in the Interaction Meshes. If desired, the user may add extra constraints such as anchoring the bodies at the feet. The approach is efficient, allowing real-time control of characters in virtual environments. Specifically, the computational cost increases only linearly in the number of frames and the complexity of the articulated body structures.

The Interaction Mesh is useful for synthesizing motions for films, computer games and digital mannequin systems. We demonstrate its usefulness in character animation by retargeting captured human motions to characters of very different proportions and volumes, such as a monkey and also by editing the motions of multiple characters while preserving the original context of the scene.

5.1.1 Contributions

- We introduce a new representation called the Interaction Mesh for encoding the spatial relationships between closely interacting body parts of articulated char-

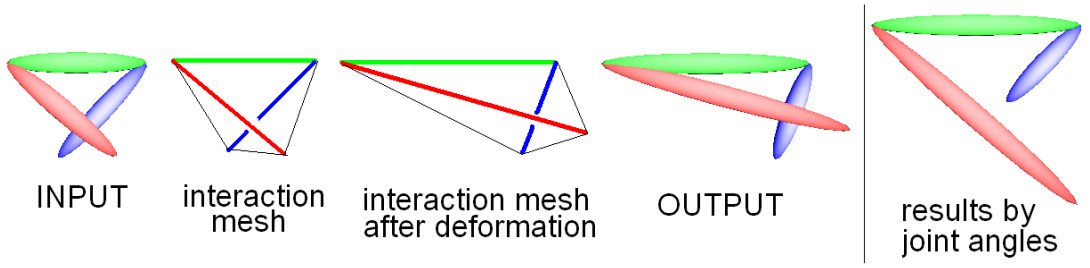


Figure 5.2: The posture of an articulated body retargeted to a new morphology with longer red/green and shorter blue segments. Note that a naïve approach by joint angles results in a change of context.

acters and objects in environment.

- We then present an automatic method that uses the Interaction Mesh for editing or retargeting motions with close interactions. The synthesized motions preserve the spatial relationships, and thus the scene semantics, while reducing the number of inappropriate interpenetrations.

5.2 Overview

We give an overview of our method in this section. First, the data of the original characters and the motion is loaded into our system. Each body segment of the character model is surrounded by a bounding volume which will be used for collision detection. The Interaction Mesh is then computed for every time frame.

Next, the user edits or retargets the motion by specifying some of the following: the target body sizes, morphology, target positions or new trajectories of some body parts. These constraint parameters are then interpolated and used to morph the original bodies to the target bodies. At every morph-step, the entire motion is adapted towards achieving the target motion. This iterative approach is necessary since the collisions between the body parts/objects need to be carefully monitored and resolved.

At every morph-step, the system adapts the motion by minimizing the Laplacian deformation of the Interaction Meshes at all the animation frames (or a fixed window of frames at a time, according to available computing resources) and the acceleration of the bodies in these frames (see Figure 5.2 for an example of the adapted result of one frame). This spacetime optimization is performed to ensure temporal coherence of the motion. The optimization is subject to various constraints, namely, bone-length constraints, collision constraints and positional constraints. Collisions are then detected

between the bounding volumes. If collisions are detected, the penetration depths are evaluated and a new set of collision constraints are defined to resolve the penetrations in the next morph-step.

5.3 Interaction Mesh

In this section, we describe how we compute the Interaction Meshes for a given motion. We assume that the mesh characters are rigged with skeletons, and each body segment is bounded by a volume (we use capsules and boxes in our experiments).

The postures of the characters are represented by the positions of the joints, rather than the joint angles. Using the joint positions as parameters has the advantage of making the constraint matrix sparse since the joints are treated as independent particles. In contrast, using the joint angles as parameters makes the Jacobian matrix very dense, as the joints near the root affect the movements of all the joints below in the hierarchy [Shi et al., 2007].

We compute the volumetric Interaction Mesh for every animation frame. Using the positions of joints and vertices of objects as a point cloud, we apply Delaunay tetrahedralization [Si and Gaertner, 2005] (see Figure 5.2). Note that the spatial relationships which we want to preserve are those between body parts that are in close proximity and are not occluded by other parts. Since the Delaunay tetrahedralization favors connecting such parts with edges, the Laplacian coordinates of vertices which are defined by vertex neighborhood will lead to mutual influence between these body parts. By the nature of Laplacian mesh editing in preserving local details, the spatial relationships of our interest will be maintained.

The orientation of some body segments cannot be computed only from the positions of joints bounding that segment. For example, the joint positions of the elbow and the wrist are insufficient to confirm the rotation around the forearm. In order to compute such orientations, we sample one extra virtual vertex on the surface of each bounding volume, as in [Shi et al., 2007]. These additional virtual vertices are added to the point cloud when defining the Interaction Mesh. Since each virtual vertex is not rigidly constrained to the body parts, its position is brought back to the original local coordinate frame once the bone's orientation is confirmed. Another possible solution is to use inverse kinematics. The orientation of the body segments can be inferred from the joint positions and the orientation in the original motion [Bodenheimer et al., 1997]. Such an approach can keep the number of vertices in the Interaction Mesh low.

5.4 Spacetime Deformation

In this section we present the spacetime optimization problem that we solve to adapt the motion at each morph-step. The spatial relationships of the body parts/objects are preserved by minimizing the Laplacian deformation energy of all the Interaction Meshes [Alexa, 2003, Zhou et al., 2005] subject to constraints derived from the morphed body sizes, detected collision and user-defined position constraints. We also introduce an acceleration energy to reduce jaggedness between frames.

5.4.1 Deformation energy

Let m be the number of vertices in the Interaction Mesh, $\mathbf{p}_j^i (1 \leq j \leq m)$ be the vertices at frame i, $\mathbf{V}_i$ be a vector of size $3m$ that includes all $\mathbf{p}_j^i$ such that $\mathbf{V}_i = (\mathbf{p}_1^{i\,\mathsf{T}}, \cdots, \mathbf{p}_m^{i\,\mathsf{T}})$, and $\mathbf{p}_j^{i\,\prime}$ and $\mathbf{V}_i'$ be the updated vectors after the deformation. The deformation energy of the mesh is defined as

$$E_L(\mathbf{V}_i') = \sum_j \frac{1}{2} \|\delta_j - L(\mathbf{p}_j^{i\,\prime})\|^2 \tag{5.1}$$

$$= \frac{1}{2}\mathbf{V}_i'^{\mathsf{T}}\mathbf{M}_i^{\mathsf{T}}\mathbf{M}_i\mathbf{V}_i' - \mathbf{b}_i^{\mathsf{T}}\mathbf{M}_i\mathbf{V}_i' + \frac{1}{2}\mathbf{b}_i^{\mathsf{T}}\mathbf{b}_i \tag{5.2}$$

where L is the operator to compute the Laplacian coordinates from the vertex locations $\mathbf{V}_i$, δ_j is the original Laplacian coordinate, and $\mathbf{M}_i, \mathbf{b}_i$ are the matrix and vector, respectively, computed by expanding Equation (5.1). The Laplacian coordinates are calculated by:

$$L(\mathbf{p}_j) = \mathbf{p}_j - \sum_{l \in N_j} w_l^j \mathbf{p}_l \tag{5.3}$$

where N_j is the one-ring neighborhood of $\mathbf{p}_j$ and w_l^j are the normalized weights which are set as inversely proportional to the distance between the vertices so that farther apart vertices have less influence on each other.

5.4.2 Acceleration energy

To reduce jaggy jumps between frames, we introduce an acceleration energy term E_A which imposes temporal relations between corresponding vertices in adjacent frames. Specifically, to reduce sudden acceleration, we minimize the movement of the corresponding vertices in adjacent frames:

$$E_A(\mathbf{V}_{i-1}', \mathbf{V}_i', \mathbf{V}_{i+1}') = \frac{1}{2}\|\mathbf{V}_{i-1}' - 2\mathbf{V}_i' + \mathbf{V}_{i+1}'\|^2 \tag{5.4}$$

where $\mathbf{V}_i'$ is the set of vertices at frame i.

5.4.3 Constraints

Here we explain the bone-length constraints, positional constraints and collision constraints imposed in the spacetime deformation.

5.4.3.1 Bone-length constraints

We introduce the bone-length constraints in order to morph the bone lengths (distance between adjacent joints) from the original scales to the target scales. In each morph-step and each animation frame, the target length l_e for each bone e is computed by linearly blending the original and final lengths. Then, a constraint enforcing the target length is imposed as $(\|\mathbf{p}_e^{1\prime} - \mathbf{p}_e^{2\prime}\| - l_e)^2$ where $\mathbf{p}_e^{1\prime}, \mathbf{p}_e^{2\prime}$ are the end vertices of the edge e. Linearizing all the bone-length constraints results in

$$C_B(\mathbf{V}_i') = \mathbf{B}_i\mathbf{V}_i' - \mathbf{l}, \tag{5.5}$$

where $\mathbf{B}_i$ is the Jacobian matrix and $\mathbf{l}$ is a vector of constant terms.

Sometimes the bone-length constraints may conflict with the Laplacian deformation energy which means satisfying the bone-length constraints increases the Laplacian deformation energy. This conflict is the main source of slow convergence. We cope with this problem by excluding the vertex $\mathbf{p}_b$ from the neighborhood of vertex $\mathbf{p}_a$ when computing the Laplacian coordinate of $\mathbf{p}_a$ if the edge connecting $\mathbf{p}_a$ and $\mathbf{p}_b$ corresponds to a bone of the body.

5.4.3.2 Positional constraints

The user can add positional constraints by anchoring some joints or a linear combination of their locations. The original trajectories of these body parts are gradually morphed to the given trajectories in each morph-step. We compute the target locations of these parts, $\mathbf{P}_i$, using linear interpolation, and write the positional constraints as:

$$C_K(\mathbf{V}_i{}') \quad = \quad \mathbf{K}_i\mathbf{V}_i' - \mathbf{P}_i \tag{5.6}$$

where $\mathbf{K}$ is a $3k \times 3m$ weight matrix that defines the influence of each joint in each positional constraint, and k is the number of positional constraints.

5.4.3.3 Collision constraints

The collision constraints prevent penetration between the bounding volumes of the skeleton. We perform collision detection by applying the ODE library [Smith, 2007]

to the current configuration of the bounding volumes. Specifically, when a penetration is detected, we compute the penetration depth, directions and the point pair penetrating each other the farthest and add the following constraints:

$$C_C(\mathbf{V}_i') = \mathbf{J}_i\mathbf{V}_i' - \mathbf{d}_i \tag{5.7}$$

where $\mathbf{J}_i$ is the Jacobian of the positions of the colliding parts with respect to the joint positions, and $\mathbf{d}_i$ is the penetration depth multiplied to the normal vectors of the penetrated surface. The Jacobian is computed by finite differencing. The joint vertices are moved and the locations of the penetrating points are recomputed according to the posture. We do not apply collision detection to adjacent body parts along the body tree structure as self-penetrations easily happen when the joints are bent.

5.4.3.4 Constraint energy

We separate the constraints in Equation (5.5)-5.7 into soft and hard constraints:

$$\mathbf{F}_i\mathbf{V}_i' = \mathbf{f}_i \ \text{(soft)}, \quad \mathbf{H}_i\mathbf{V}_i' = \mathbf{h}_i \ \text{(hard)} \tag{5.8}$$

and define a constraint energy that represents the amount of violation of the soft constraints:

$$E_C(\mathbf{V}_i') \quad = \quad \frac{1}{2}\mathbf{V}_i'^{\mathsf{T}}\mathbf{F}_i^{\mathsf{T}}\mathbf{W}\mathbf{F}_i\mathbf{V}_i' - \mathbf{f}_i^{\mathsf{T}}\mathbf{W}\mathbf{F}_i\mathbf{V}_i' + \frac{1}{2}\mathbf{f}_i^{\mathsf{T}}\mathbf{W}\mathbf{f}_i. \tag{5.9}$$

where $\mathbf{W}$ is a square diagnol matrix that assigns a different weight to each constraint.

By default, we set the bone-length constraints and one positional constraint (supporting foot) hard, and the collision constraints and the rest of the positional constraints soft. The bone-length constraints are set hard so that the bodies are correctly scaled to the target values. Soft collision constraints stabilize the motion when there is little open space. It also provides the animator some results when collisions cannot be avoided due to insufficient open space when bodies are enlarged. It is also necessary to set the other positional constraints soft to avoid over constraining. By default, the weights in $\mathbf{W}$ are set 4.0 and 0.4 for the collision and additional positional constraints, respectively. The constraints can be switched between soft and hard according to the desired animation effect. When the number of morph steps is small, we also need to set the bone-length constraints soft, and their weights are set to 2.0.

5.4.4 Iterative Morphing

At every morph-step, the body sizes and the positional constraints are updated, and the motions of the characters are adapted by minimizing the sum of the deformation (Eq.5.1), acceleration (Eq.5.4) and constraint energy (Eq.5.9) of all frames subject to the hard constraints. The adapted motion is computed by solving

$$\arg\min_{\mathbf{V}'_i,\lambda_i(1\leq i\leq n)} \sum_i^n E_L + w_\Delta E_A + E_C + \lambda_i^\intercal(\mathbf{H}_i\mathbf{V}'_i - \mathbf{h}_i) \tag{5.10}$$

where n is the number of frames, $\mathbf{V}'_i$ is the set of new vertex positions at frame i, λ_i are the Lagrange multipliers and w_Δ is a constant weight (we use 0.2). Note that E_A is not defined for the first and last frames, hence we set them to zero. The spacetime optimization problem in Equation (5.10) can be solved by differentiating it with respect to $\mathbf{V}'_i$ and λ_i, and solving the following linear equation:

$$\begin{pmatrix} \mathcal{M}^\intercal\mathcal{M} + w_\Delta\mathcal{A}^\intercal\mathcal{A} + \mathcal{F}^{k\intercal}\mathcal{W}\mathcal{F}^k & \mathcal{C}^{k\intercal} \\ \mathcal{C}^k & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathcal{V} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathcal{M}^\intercal\mathcal{B} + \mathcal{F}^{k\intercal}\mathcal{W}f \\ \mathcal{H} \end{pmatrix} \tag{5.11}$$

where $\mathcal{B}$, $\mathcal{H}$, f, $\mathcal{V}$ and λ are vectors that include $\mathbf{b}_i$ (in Eq.5.2), $\mathbf{h}_i$, $\mathbf{f}_i$ (in Eq.5.8), $\mathbf{V}'_i$ and λ_i for all the frames, respectively, i.e. $\mathcal{B} = (\mathbf{b}_1^\intercal, ...\mathbf{b}_n^\intercal)^\intercal$, $\mathcal{H} = (\mathbf{h}_1^\intercal, ...\mathbf{h}_n^\intercal)^\intercal$, $f = (\mathbf{f}_1^\intercal, ...\mathbf{f}_n^\intercal)^\intercal$, $\mathcal{V} = (\mathbf{V}_1'^\intercal, ...\mathbf{V}_n'^\intercal)^\intercal$, $\lambda = (\lambda_1^\intercal, ...\lambda_n^\intercal)^\intercal$ and $\mathcal{M}$ is the Laplacian matrix, $\mathcal{F}^k$ is a soft constraint matrix at the k-th morph-step, each of which includes $\mathbf{M}_i$ in Equation (5.2), and $\mathbf{F}_i$ in Equation (5.8) for all the frames, respectively:

$$\mathcal{M} = \begin{pmatrix} \mathbf{M}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{M}_n \end{pmatrix}, \mathcal{F}^k = \begin{pmatrix} \mathbf{F}_1^k & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{F}_n^k \end{pmatrix},$$

and $\mathcal{A}$ is a matrix that computes the acceleration for all the frames from $\mathcal{V}$, $\mathcal{W} = \mathrm{diag}(\mathbf{W}, ..., \mathbf{W})$, and $\mathcal{C}^k$ is the constraint matrix at the k-th morph-step, which includes $\mathbf{H}_i$ in Equation (5.8) for all the frames:

$$\mathcal{A} = \begin{pmatrix} \mathbf{0} & & & & \\ \mathbf{I} & -2\mathbf{I} & \mathbf{I} & & \\ & & \ddots & & \\ & & \mathbf{I} & -2\mathbf{I} & \mathbf{I} \\ & & & & \mathbf{0} \end{pmatrix}, \mathcal{C}^k = \begin{pmatrix} \mathbf{H}_1^k & \cdots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{H}_n^k \end{pmatrix}.$$

Algorithm 4 Motion Adaptation by Interaction Mesh

Input: Skeleton and input motion sequence

initial / final body scaling factors: $\mathbf{s}_0, \mathbf{s}_f$

locations of initial/final positional constraints: $\mathbf{p}_0, \mathbf{p}_f$

Output: Target motion sequence

(Initialization)

- Compute the Interaction Meshes (vertex positions and connectivity) of all input frames.

(Motion Adaptation)

for k=1 to N morph-steps **do**

- Update body scale / location of positional constraints:

$$\mathbf{s}_k \leftarrow \frac{N-k}{N}\mathbf{s}_0 + \frac{k}{N}\mathbf{s}_f, \quad \mathbf{p}_j \leftarrow \frac{N-k}{N}\mathbf{p}_0 + \frac{k}{N}\mathbf{p}_f$$

- Update the constraint matrices $\mathbf{F}_i, \mathbf{H}_i$, the target values of the constraints $\mathbf{f}_i, \mathbf{h}_i$ in Equation (5.8) and the deformation vector $\mathbf{b}_i$ in Equation (5.2) for frames $i = 1, ..., n$.

- Update the vertex locations by solving Equation (5.10).

- Compute segment orientations and update virtual vertices.

end for

Our motion adaptation algorithm is summarized in Algorithm 4.

The Interaction Meshes are defined for the original motion frames and their connectivities are kept unchanged at all the morph-steps, which results in a constant $\mathcal{M}$. Note that re-computing the tetrahedralization at each morph-step would result in gradual drifting of the motion away from the original sequence. Keeping the mesh topology from the original motion helps to preserve the spatial relationships of the components in the original motion.

5.4.5 Possible artifacts and solutions

Due to the non-uniform weighting of the edges, drastic updates of the morphing parameters such as the character sizes may result in flipping of the tetrahedra in the Interaction Meshes. Such flipping, if it occurs in an open space does not cause noticeable artifacts, but may result in a change of context if it happens with a tetrahedron composed of two bones which are nearby but not adjacent to each other. We prevent the flipping by detecting collisions between the body parts and applying the collision constraints (Eq.5.7) at each morph step. However, the system may fail when the lin-

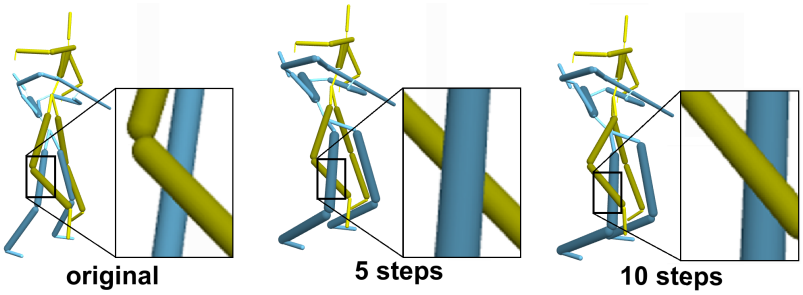


Figure 5.3: Flipping of a tetrahedron during a morph-step may lead to a change of spatial context: (left) initial posture, (middle) flipping happens at the left lower leg of the yellow character when using 5 morph-steps, (right) flipping avoided when using 10 morph-steps.

earization of the collision constraints breaks down, which happens when the body sizes are very narrow and the penetration depth is too large. This results in pushing out the penetrating body in the wrong direction (see Figure 5.3). The problem can be avoided by giving enough volume to the body parts and setting small morph steps. We obtained satisfactory results using 10 morph-steps for all the examples.

The soft collision constraints can result in penetrations when the bodies are enlarged too much while there is limited open space. We can alert the user of the lack of space by giving the sum of the squares of penetration depth for reference. The user can then choose to either enlarge the environment or stop scaling.

The system may become unstable when the original motion contains movements where the body parts pass through each other. When this happens, the direction the collision constraint pushes the bodies away from each other will turn opposite at some moment, resulting in a large movement in one frame. This contradicts the minimization of acceleration energy and causes vibrations. Switching off the collision constraints at the frames of these pass throughs can remove such artifacts.

5.5 Experimental Results

In this section, we show experimental results from applying our motion retargeting method to character animation. We apply it to several types of motions with close interactions, namely, between body parts of a single character or multiple characters, and between a character and its environment. We also demonstrate its usefulness for real-time character control. The heights of the feet are all constrained to the original values by hard constraints by default, which are necessary especially when the characters are

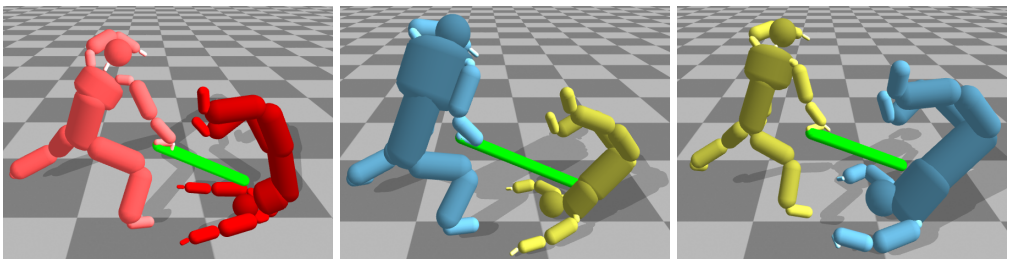


Figure 5.4: Snapshots of a sword attacking motion (left) retargeted to characters of different morphologies (middle, right).

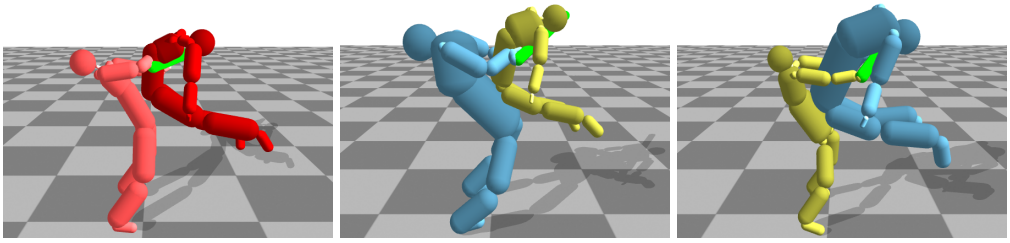


Figure 5.5: Snapshots of a back breaking attacking motion (left) retargeted to characters of different morphologies (middle, right).

standing on the ground.

5.5.1 Retargeting Motions of Close Interactions

First we show the results of retargeting motions involving only characters. We use motions involving much tangling and contacts, such as judo, as well as those with few contacts, such as dancing and attackers/defenders in fighting games.

5.5.2 Judo attacks

Our first motion example is a judo "Ogoshi throw" in which the attacker holds the arm and the waist of the defender and throws the defender by carrying him/her onto the back. We retarget the motions of the thrower and the defender to characters of various morphologies. We use the body sizes of characters in Allen et al. [Allen et al., 2003] as reference (See Figure 5.1). Here, although the proportions and the bounding volumes of the new characters are completely different from those of the original characters, our system can still produce the Ogoshi throw. Note that previous motion editing/retargeting approaches are difficult to apply to this kind of close interactions since they only consider the joint angles of the original motion but not the spatial relationships. As a result, the retargeted motion may have a different context (e.g. the

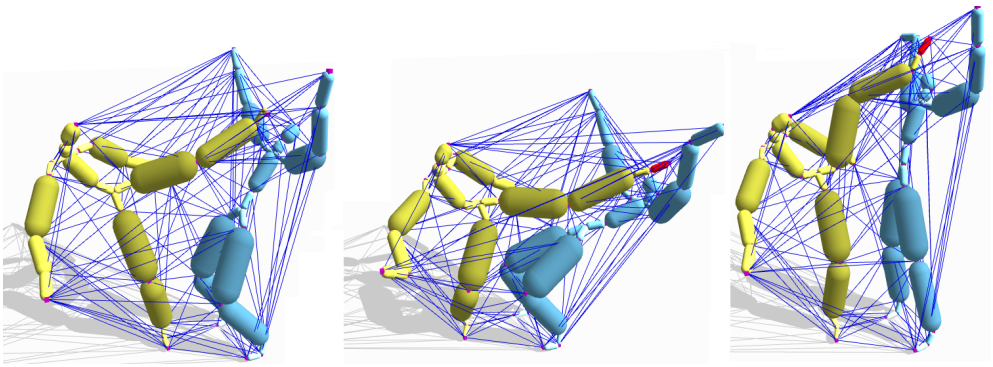


Figure 5.6: (left) A posture of a turn kick interaction. (middle, right) The animator drags the left foot of the yellow character by mouse and the other character moves to preserve the spatial relationship.

arms extend to the other side of the defender's body). They also require the animators to manually specify all the positional constraints in all the frames, such as the attacker's hand holding the defender's body. This can be a tedious task for the animator.

5.5.3 Fighting scenes

Our next two motion examples are fighting scenes involving two characters provided by a game company.

The first scene involves a character holding a sword attacking its enemy. The sword penetrates through the enemy character when the enemy is stabbed, and therefore, we turned off the collision constraint in those frames. The second scene is from the same game. The character holding the sword breaks the back of the enemy with both arms and drops it. The sword unintentionally passes through the arm of the defender in some of the frames, which is due to the manual design. Again, we turned off the collision constraint in those frames. Both the attacker's and defender's morphologies are changed and animations of different combination of bodies are created. Snapshots of the original and synthesized motions are shown in Figure 5.4 and Figure 5.5, respectively. Note that manual editing would require a lot of care to avoid unintentional penetrations of the sword into the body. Our method can automatically produce the motion of passing the sword into the space between the enemy's arm and torso. These examples show that our method can also be used for manually designed motions which are not penetration-free. In such cases, the context of the penetrations (e.g. stabbing the enemy) are preserved in the synthesized results. Feedback from the game company indicates that the quality of our retargeted movements is high enough for games usage.

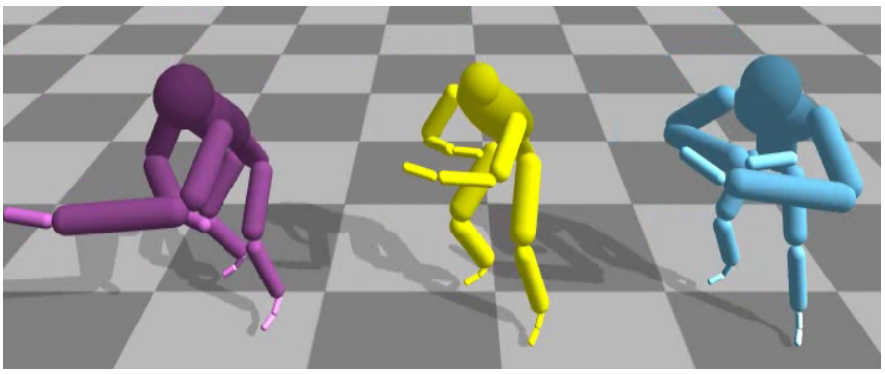


Figure 5.7: Original dancing motion (middle) and the retargeted results to a monkey model with long arms using a joint-angle based method (left) and using our method (right).

5.5.4 Interactive character control

Next, we show a demo of using the Interaction Mesh for real-time control of characters. Pausing the animation of interaction at some frame, we can let the user interactively control a body part using inverse kinematics while maintaining its spatial relationships with the other character(s). In such cases, we solve Equation (5.11) for a single frame rather than for the entire motion, which provides real-time performance. The other character(s) will follow the movements of the controlled character according to the Interaction Mesh at that frame. The controlled body part is softly constrained by an additional positional constraint. An example of editing a posture in a turn-kick motion is shown in Figure 5.6. The updates in the edited frame can be propagated to the whole motion by iteratively solving Equation (5.10) using the edited posture as a constraint.

5.5.5 Single character motions

We use a dancing motion in which the character performs an arms cycling motion and retarget it to a character with monkey proportions (Figure 5.7). Our method can preserve the context of the motion despite the much longer arms of the monkey character. In contrast, the method of [Lyard and Magnenat-Thalmann, 2008] results in a motion with many collisions, causing the movements to appear unstable. Note that since this motion does not involve any tangles, the topology coordinates presented in Chapter 4 are also difficult to apply.

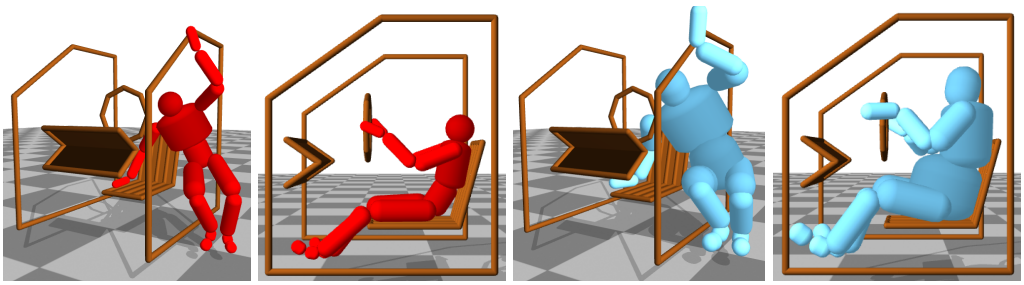


Figure 5.8: Snapshots of a character getting into and riding a car model; (red) original character and (blue) a tall fat character.

5.5.6 Motion Adaptation in a Constrained Environment

Motions in a constrained environment, such as getting in and out of a car, involve close maneuvers and collision avoidance. Retargeting such motions to characters of different sizes or adapting the motions to environment with different parameters (e.g. size of car) has a great demand in CAD design and digital mannequins [Badler et al., 1999]. Here we show examples of using the Interaction Mesh for such a purpose. We captured the motion of a person getting into a car and holding the steering wheel. The environment is composed of simple polylines representing the car doors, ceiling, floor, seats, and steering wheel. The Interaction Mesh is composed of the vertices of the environment and the skeleton joints and end effectors. Snapshots of the input motion and that retargeted to a scaled character are shown in Figure 5.8. Observe that the character's motion is successfully adapted to the new character size. Since some of the interactions of the character with the car, such as ducking and passing through the narrow space, cannot be described only with explicit constraints such as contacts, these motions are difficult to handle for previous methods.

5.5.7 Computational Costs

The main bottleneck of our method is solving the large linear equation in Equation (5.11). We use UMFPACK [Davis, 2004] and GotoBLAS [Goto and Van De Geijn, 2008]. Since the Laplacian matrix $\mathcal{M}$ and the constraint matrix $\mathcal{C}^k$ are both sparse, the computation only increases linearly with respect to the number of vertices in all the Interaction Meshes. Therefore, the complexity of the problem is $O(m \times n)$, where m is the number of vertices in each mesh and n is the number of frames. For all the retargeting examples shown in this research, the computation required for each motion is around 1 minute for an animation of 100 frames, using one core of a Core i7 2.67GHz CPU. Since most

of the computation, such as the computation of the Laplacian and constraint matrices and solving the large linear system [Bolz et al., 2003] are highly parallelizable, much faster response can be expected with GPU implementations.

5.6 Discussions

In this section, we compare our approach with previous methods in terms of advantages and limitations.

Simply combining the kinematic constraints and collision avoidance as is done in [Lyard and Magnenat-Thalmann, 2008] cannot preserve spatial relationships between bones. Although collision detection can avoid interpenetration of body parts, the movement of one part does not affect the movements of the other parts until collision occurs. Moreover, as shown in Figure 5.7, left, the collision does not necessarily repulse the body in the direction that maintains the context of the original scene. As a result, coordination between the bodies can easily be lost. With our method, the Interaction Mesh moves all the nearby bones together such that their spatial relationships are maintained.

Shi et al. [Shi et al., 2007] apply a cascading scheme for motion retargeting to speed up the process and make use of the GPU resources. Such a cascading scheme is inapplicable to the motions of our interest since it may greatly update the posture after one iteration. Large updates may be acceptable for open postures such as standing and reaching out for an object, however for motions with close interactions between body parts, resolving all the collisions after a large update may result in a change of context, for example, the body parts may be flipped to the opposite side. This also explains why we need to gradually morph the morphology of the characters and sizes of objects in the scene.

The concept of the Interaction Mesh is very general. It is possible to compute the Interaction Mesh using vertices of freeform meshes and use it for shape deformation. This can be an interesting application of deformation transfer [Sumner and Popovic, 2004, Zayer et al., 2005] and cloth animation. For example, animation of a character wrapping an object or wearing clothes.

Using a high resolution volumetric mesh, such as one defined by the surface of a mesh character, may be a good solution for intricate finger motions, such as manipulating a thread or small object, as it can represent the details of the interaction between the fingers' surface and the object. However, applying such a method for spacetime

control of a full body character will be very computationally costly due to the huge number of vertices.

Spacetime optimization is known to be an important tool for creating realistic character motions since it allows the characters to prepare for the interactions ahead along the timeline [Liu et al., 2006], such as an early bending of the torso before entering a car. It also removes jaggedness from the movements, which is a drawback of frame-based approaches. Therefore, we argue that our combined use of a volumetric structure defined by joint positions and the spacetime optimization is a good design choice for efficient synthesis of realistic motions of interactions.

The acceleration energy term might unnecessarily smooth the large accelerations from impacts, which is often an important feature of motions. Experimentally, we have found the advantage of using the acceleration term outweigh the smoothing of high impulse movements. One possible solution for maintaining the features of high acceleration is to use inter-frame Laplacian coordinates proposed in Kwon et al. [Kwon et al., 2008b] instead of the acceleration energy.

5.6.1 Limitations

Our method may fail when the constraints are drastically different from those in the original motion, such as when one of the interacting characters is scaled too small or too large. In such situations, the body parts may be too far apart to maintain the interactions. However, any methods that use inverse kinematics are not immune to such a problem under extreme scaling. An example of this vulnerability is when a character is in contact with another object at many parts of its body, for example, a full body hold. If the object's size is scaled down, the Laplacian deformation will try to preserve the spatial relationship at all the contact areas, which is physically impossible due to the rigidity of the body parts. As a result, the deformation error will be shared among all the contact areas, resulting in the loss of all contacts. This is related to the fact that our method does not require user-specified contact constraints. A simple possible solution is to prioritize such contacts and impose contact constraints at areas which appear to be important. Another solution may be to recompute the Interaction Meshes at certain morph-steps when the spatial relationship is difficult to preserve and allow the mesh to drift from the original topology.

5.7 Conclusion and Future Work

In this research, we have presented a new method to edit and retarget character animation that involves many close interactions by introducing a new representation called Interaction Mesh. When updating the motion, the spatial relationships between different body components and objects can be preserved by applying a spacetime optimization to minimize the deformation of the Interaction Meshes subject to collision constraints as well as bone-length and positional constraints. The method is fully automatic, not requiring manual intervention from the user. We have demonstrated the effectiveness of the proposed method by showing realistic synthesized animations of various types of close interactions, which are difficult to produce using previous methods.

As future work, we plan to numerically evaluate the topological and geometric features of the Interaction Meshes, and introduce a metric to compare motions at the level of Interaction Meshes. Specifically, we are interested in parameterizing the Interaction Mesh. We will then be able to evaluate the similarity of motions. As a result, the parameterized Interaction Mesh can be applied to various applications such as motion blending and concatenation in Motion Graphs [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002]. We will also be able to extend our method for use in reinforcement learning, enabling computer-controlled characters to smartly interact with user-controlled characters in closely interacting environments.

Chapter 6

Conclusion

6.1 Summary and Discussion

In this thesis, we have presented new representations for encoding the spatial relationships between multiple characters and objects in proximity. Experimental results show that the new representations can be applied to various problems including motion editing and retrieval in character animation.

6.1.1 Human Motion Indexing and Retrieval

Most of the previous research on human motion indexing and retrieval compute the Euclidean distance of joint angles or joint positions. When retrieving human motions, we are more interested in the context of the scene rather than low level representations such as joint angles. The work in [Müller et al., 2005, Müller and Röder, 2006] extract the semantics of the motions using a manually predefined boolean feature set to encode the relationships between joints. Semantically similar motions can be retrieved using such an approach. However, users have to specify all the boolean features that define the semantics of the scene in advance.

We proposed TangleList to encode the tangles made between the body parts of single / multiple characters. Using Gauss Linking Integral, we can directly compute the features which are robust against minor kinematic differences. The TangleList contains the semantics of the scene as each element describes how each body part is tangled with the others. We further defined a distance metric based on TangleList. Experimental results show that the new metric is also useful for finding motions which can be blended without any penetrations.

6.1.2 Motion Synthesis

Existing methods for creating charcter animations, such as Motion Graphs [Arikan and Forsyth, 2002, Lee et al., 2002, Kovar et al., 2002], Motion Retargetting [Gleicher, 1998] and Keyframe animations, are based on low level representations such as joint angles and positions. Interpolating postures at such a level will result in a huge amount random sampling and collision detection [Lavalle and Kuffner, 2000], especially when the movement involves close interactions of characters. However, the computational cost increases exponentially when the number of DOFs increases. When handling close interactions between multiple characters, the huge computational cost makes it infeasible for real-time applications such as 3D computer games.

The concept of Topology Coordinates presented in Chapter 4 is an abstract representation that describes the spatial relationship between the tangled body parts. Using Topology Coordinates, we can directly plan and edit close interactions in motion space. This allows us to use local search methods instead of global path-planning algorithms. A smooth transition in the motion space which has no interpenetration of the body parts can be obtained by linearly interpolating the Topology Coordinates from two different postures/keyframes. Experimental results show that our method is significantly more efficient than global path-planning algorithms for complex motions such as tangling the segments of the body.

6.1.3 Motion Adaption and Deformation Transfer

We further proposed a new representation called Interaction Mesh to handle close interactions without many tangles. While Topology Coordinates are computed based on 1D links, the Interaction Mesh can encode the spatial relationship between 2D manifolds such as meshed characters, cloth models and 3D environments composed of polygons. In addition, we proposed a unified framework to automatically compute and maintain the spatial relationship between characters and objects in the scene when editing and retargetting the motions.

6.2 Review of Contributions

The contributions of this thesis are summarized as follows:

- We proposed a new topology-based representation for describing the relationship between two characters who are closely interacting with each other. The

relationship of two bodies are computed by Gauss Linking Integral and encoded by rational tangles.

- We proposed the concept of *Topology Coordinates*, in which the spatial relationships of the segments are embedded. We can edit and plan motions of closely interacting characters by local search which significantly reduce the computation for collision avoidance.

- We proposed the concept of *Interaction Mesh*, which encodes the spatial relationships between closely interacting body parts of articulated characters and objects in the environment. We further propose an automatic method for editing / retargeting close interactions of characters.

6.3 Directions for Future Work

In this section, we will discuss the future directions for extending the work presented in this thesis. First of all, we would like to improve the performance of our motion synthesis methods by using parallel programming techniques. General-purpose computing on graphics processing units (GPGPU) takes the advantages of using high-performance multi-core processors on graphics card and the highly parallelizable nature of GPU programming for computations which are traditionally performed by CPUs. Since most of the computation, such as the computation of the Laplacian and constraint matrices and solving the large linear system, in our proposed frameworks are highly parallelizable, much faster response can be expected with GPU implementations. Then we will be able to handle large scale problems such as adapting the motions of many characters and objects at interactive frame rate.

Secondly, we edit the postures/motions of the characters kinematically in the proposed methods. In the future, we would like to apply our method for controlling characters in physically-based environments, such as by combining it with the balance keeping techniques in [da Silva et al., 2008, Macchietto et al., 2009]. We cannot directly apply our methods to control robot as physical constraints and contact force are not included in our framework. Tackling such a problem may also lead to solutions for controlling multi-biped robots to cooperatively accomplish tasks such as carrying objects together. With the future direction for improving the performance of our frameworks discussed in last paragraph, we believe that the physical-based interaction editing method can be applied for real-time applications such as 3D computer games

and controlling humanoids interactively.

Finally, we would like to further generalize the concept of topology-based motion editing for 2D surfaces, such that we can handle cloths and meshed characters. Using the Interaction Mesh can be a solution. However, when handling large scale problems such as the interaction of high resolution cloth, we are interested in the topology of the cloths rather than the local spatial relationships between all vertices. An interesting research direction would be representing the spatial relationship of 2D manifolds by an abstract representation like the concept of using Topology Coordinates for 1D strands. For example, the winding number is a signed number which computes how many times a point is surrounded by a 2D curve in the counter-clockwise direction. This is an abstraction of the representation between the two interacting objects. We are interested in extending the concept of the winding number for 2D manifolds. By doing this, we will be able to represent how the 2D manifolds are facing each other. As a result, we can make use of the extended version of winding number for indexing and editing motions of meshed characters as well as deformable objects such as cloth.

Bibliography

[Adams, 1994] Adams, C. C. (1994). *The Knot Book*. W. H. Freeman and Co.

[Agarwal et al., 2002] Agarwal, P. K., Edelsbrunner, H., and Wang, Y. (2002). Computing the writhing number of a polygonal knot. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 791–799, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[Alexa, 2003] Alexa, M. (2003). Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2-3):105–114.

[Alexa et al., 2000] Alexa, M., Cohen-Or, D., and Levin, D. (2000). As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Allen et al., 2003] Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03*, pages 587–594.

[Arikan and Forsyth, 2002] Arikan, O. and Forsyth, D. A. (2002). Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490.

[Arikan et al., 2003] Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2003). Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408.

[Arikan et al., 2005] Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2005). Pushing people around. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 59–66, New York, NY, USA. ACM Press.

[Badler et al., 1999] Badler, N. I., Palmer, M. S., and Bindiganavale, R. (1999). Animation control for real-time virtual humans. *Communications of the ACM*, 42(8):64–73.

[Baraff, 1996] Baraff, D. (1996). Linear-time dynamics using lagrange multipliers. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 137–146, New York, NY, USA. ACM.

[Bertram et al., 2006] Bertram, D., Kuffner, J., Dillmann, R., and Asfour, T. (2006). An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1874–1879. IEEE.

[Bodenheimer et al., 1997] Bodenheimer, B., Rose, C., Rosenthal, S., and Pella., J. (1997). The process of motion capture: Dealing with the data. In *Computer Animation and Simulation 97*, pages 3–18.

[Bolz et al., 2003] Bolz, J., Farmer, I., Grinspun, E., and Schröoder, P. (2003). Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Transactions on Graphics*, 22(3):917–924.

[Cohen, 1992] Cohen, M. F. (1992). Interactive spacetime control for animation. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 293–302, New York, NY, USA. ACM.

[Conway, 1970] Conway, J. H. (1970). An enumeration of knots and links, and some of their algebraic properties. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 329–358, Oxford. Pergamon, Press.

[da Silva et al., 2008] da Silva, M., Abe, Y., and Popović, J. (2008). Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics*, 27(3):82:1–82:10.

[Davis, 2004] Davis, T. A. (2004). Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2 (Jun.)):196–199.

[Electronic Arts. Inc., 2006] Electronic Arts. Inc. (2006). Nba live 06.

[Erdmann, 2004] Erdmann, M. A. (2004). Protein similarity from knot theory and geometric convolution. In *RECOMB '04: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 195–204, New York, NY, USA. ACM.

[Fang and Pollard, 2003] Fang, A. C. and Pollard, N. S. (2003). Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics*, 22(3):417–426.

[Gleicher, 1997] Gleicher, M. (1997). Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA. ACM Press.

[Gleicher, 1998] Gleicher, M. (1998). Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, New York, NY, USA. ACM Press.

[Goto and Van De Geijn, 2008] Goto, K. and Van De Geijn, R. (2008). High-performance implementation of the level-3 blas. *ACM Transactions on Mathematical Software*, 35(1):1–14.

[Graepel et al., 2004] Graepel, T., Herbrich, R., and Gold, J. (2004). Learning to fight. *Proceedings of Computer Games: Artificial Intelligence Design and Education (CGAIDE 2004)*, pages 193–200.

[Grochow et al., 2004] Grochow, K., Martin, S. L., Hertzmann, A., and Popović, Z. (2004). Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531.

[Heck and Gleicher, 2007] Heck, R. and Gleicher, M. (2007). Parametric motion graphs. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 129–136, New York, NY, USA. ACM Press.

[Ho and Komura, 2007a] Ho, E. S. and Komura, T. (2007a). Planning tangling motions for humanoids. In *Proceedings of the 7th IEEE-RAS International Conference on Humanoid Robots, 2007*, pages 507–512.

[Ho and Komura, 2007b] Ho, E. S. and Komura, T. (2007b). Wrestle alone: Creating tangled motions of multiple avatars from individually captured motions. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 427–430, Washington, DC, USA. IEEE Computer Society.

[Ho and Komura, 2009a] Ho, E. S. and Komura, T. (2009a). Character motion synthesis by topology coordinates. In Dutr'e, P. and Stamminger, M., editors, *Computer Graphics Forum (Proceedings of Eurographics 2009)*, volume 28, pages 299–308, Munich, Germany.

[Ho and Komura, 2009b] Ho, E. S. and Komura, T. (2009b). Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):481–492.

[Ho and Komura, 2009c] Ho, E. S. and Komura, T. (2009c). Real-time character control for wrestling games. *Springer Lecture Notes in Computer Science (Motion in Games 2009)*, LNCS 5884:128–137.

[Ho and Komura, 2010] Ho, E. S. and Komura, T. (2010). A finite state machine based on topology coordinates for wrestling games. *To appear in Journal of Visualization and Computer Animation*.

[Ho et al., 2010] Ho, E. S. L., Komura, T., and Tai, C.-L. (2010). Spatial relationship preserving character motion adaptation. *ACM Transactions on Graphics*, 29(4):1–8.

[Hodgins and Pollard, 1997] Hodgins, J. K. and Pollard, N. S. (1997). Adapting simulated behaviors for new characters. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 153–162, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Hodgins et al., 1995] Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating human athletics. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA. ACM.

[ILOG Inc, 2005] ILOG Inc (2005). *ILOG CPLEX 9.1 User's and reference manual*.

[Kagami et al., 2000] Kagami, S., Kanehiro, F., Tmiya, Y., Inaba, M., and Inoue, H. (2000). Autobalancer: An online dynamic balance compensation scheme for humanoid robots. pages 329–340. A K PetersLtd.

[Kauffman and Lambropoulou, 2003] Kauffman, L. H. and Lambropoulou, S. (2003). On the classification of rational tangles.

[Kauffman and Lambropoulou, 2004] Kauffman, L. H. and Lambropoulou, S. (2004). On the classification of rational tangles. *Advances in Applied Mathematics*, 33(2):199–237.

[Kavraki et al., 1994] Kavraki, L., Svestka, P., Latombe, J., and Overmars, M. (1994). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical report, Stanford, CA, USA.

[Keogh et al., 2004] Keogh, E., Palpanas, T., Zordan, V. B., Gunopulos, D., and Cardle, M. (2004). Indexing large human-motion databases. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 780–791. VLDB Endowment.

[Klenin and Langowski, 2000] Klenin, K. and Langowski, J. (2000). Computation of writhe in modeling of supercoiled dna. *Biopolymers*, 54:307–317.

[Komura et al., 2005] Komura, T., Ho, E. S., and Lau, R. W. (2005). Animating reactive motion using momentum-based inverse kinematics. *Journal of Computer Animation and Virtual Worlds (special issue of CASA 2005)*, 16(3):213–223.

[Kovar and Gleicher, 2003] Kovar, L. and Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 214–224, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Kovar and Gleicher, 2004] Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568.

[Kovar et al., 2002] Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482.

[Kwon et al., 2008a] Kwon, T., Cho, Y.-S., Park, S. I., and Shin, S. Y. (2008a). Two-character motion analysis and synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):707–720.

[Kwon et al., 2008b] Kwon, T., Lee, K. H., Lee, J., and Takahashi, S. (2008b). Group motion editing. *ACM Transactions on Graphics*, 27(3):80:1–80:8.

[Kwon and Shin, 2005] Kwon, T. and Shin, S. Y. (2005). Motion modeling for on-line locomotion synthesis. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 29–38, New York, NY, USA. ACM.

[Lau and Kuffner, 2005] Lau, M. and Kuffner, J. J. (2005). Behavior planning for character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 271–280, New York, NY, USA. ACM.

[Lau and Kuffner, 2006] Lau, M. and Kuffner, J. J. (2006). Precomputed search trees: planning for interactive goal-driven animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 299–308, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Lavalle and Kuffner, 2000] Lavalle, S. M. and Kuffner, J. J. (2000). Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308.

[LaValle and Kuffner, 2001] LaValle, S. M. and Kuffner, J. J. (2001). Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400.

[Lawrence, 2004] Lawrence, N. D. (2004). Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems (Proceedings of NIPS 2003)*, pages 329–336, Cambridge, MA. MIT Press.

[Lee et al., 2002] Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500.

[Lee and Lee, 2004] Lee, J. and Lee, K. H. (2004). Precomputing avatar behavior from human motion data. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 79–87, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Lee et al., 2007] Lee, K. H., Choi, M. G., Hong, Q., and Lee, J. (2007). Group behavior from video: a data-driven approach to crowd simulation. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer*

animation, pages 109–118, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Lerner et al., 2007] Lerner, A., Chrysanthou, Y., and Lischinski, D. (2007). Crowds by example. *Computer Graphics Forum (Proceedings of Eurographics)*, 26(3):655–664.

[Levitt, 1983] Levitt, M. (1983). Protein folding by restrained energy minimization and molecular dynamics. *J. Mol. Biol*, 170:723–764.

[Liu et al., 2005] Liu, C. K., Hertzmann, A., and Popović, Z. (2005). Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics*, 24(3):1071–1081.

[Liu et al., 2006] Liu, C. K., Hertzmann, A., and Popović, Z. (2006). Composition of complex optimal multi-character motions. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 215–222, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Liu and Popović, 2002] Liu, C. K. and Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics*, 21(3):408–416.

[Liu et al., 2003] Liu, F., Zhuang, Y., Wu, F., and Pan, Y. (2003). 3d motion retrieval with motion index tree. *Computer Vision and Image Understanding*, 92(2-3):265–284.

[Liu et al., 1994] Liu, Z., Gortler, S. J., and Cohen, M. F. (1994). Hierarchical space-time control. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 35–42, New York, NY, USA. ACM.

[Lyard and Magnenat-Thalmann, 2008] Lyard, E. and Magnenat-Thalmann, N. (2008). Motion adaptation based on character shape. *Comput. Animat. Virtual Worlds*, 19(3-4):189–198.

[Macchietto et al., 2009] Macchietto, A., Zordan, V., and Shelton, C. R. (2009). Momentum control for balance. *ACM Transactions on Graphics*, 28(3):1–8.

[Matsuno et al., 2006] Matsuno, T., Tamaki, D., Arai, F., and Fukuda, T. (2006). Manipulation of deformable linear objects using knot invariants to classify the object condition based on image sensor information. *IEEE/ASME Transactions on Mechatronics*, 11(4):401–408.

[Mukai and Kuriyama, 2005] Mukai, T. and Kuriyama, S. (2005). Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070.

[Müller and Röder, 2006] Müller, M. and Röder, T. (2006). Motion templates for automatic classification and retrieval of motion capture data. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 137–146, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Müller et al., 2005] Müller, M., Röder, T., and Clausen, M. (2005). Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics*, 24(3):677–685.

[Nishiwaki et al., 2001] Nishiwaki, K., Sugihara, T., Kagami, S., Inaba, M., and Inoue, H. (2001). Online mixture and connection of basic motions for humanoid walking control by footprint specification. *2001 IEEE International Conference on Robotics and Automation(ICRA'01)*, pages 4110–4115.

[Paris et al., 2007] Paris, S., Pettré, J., and Donikian, S. (2007). Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum*, 26(3):665–674.

[Park et al., 2004] Park, S. I., Kwon, T., Shin, H. J., and Shin, S. Y. (2004). Analysis and synthesis of interactive two-character motions. *Technical Note, KAIST, CS/TR-2004-194*.

[Park et al., 2002] Park, S. I., Shin, H. J., and Shin, S. Y. (2002). On-line locomotion generation based on motion blending. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 105–111, New York, NY, USA. ACM Press.

[Pohl, 1968] Pohl, W. (1968). The self-linking number of a closed space curve. *Journal of Mathematics and Mechanics*, 17:975–985.

[Popović and Witkin, 1999] Popović, Z. and Witkin, A. (1999). Physically based motion transformation. In *SIGGRAPH '99: Proceedings of the 26th annual conference*

on Computer graphics and interactive techniques, pages 11–20, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Raibert and Hodgins, 1991] Raibert, M. H. and Hodgins, J. K. (1991). Animation of dynamic legged locomotion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 349–358, New York, NY, USA. ACM.

[Reidemeister, 1935] Reidemeister, K. (1935). *Knot Theory*. BCS Associates, Moscow, Idaho, U.S.A.

[Rose et al., 1998] Rose, C., Cohen, M. F., and Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18:32–40.

[Safonova and Hodgins, 2007] Safonova, A. and Hodgins, J. K. (2007). Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics*, 26(3):106.

[Saha and Isto, 2006] Saha, M. and Isto, P. (2006). Motion planning for robotic manipulation of deformable linear objects. *Proceedings of IEEE International Conference on Robotics and Automation (2006)*, pages 2478–2484.

[Saha and Isto, 2007] Saha, M. and Isto, P. (2007). Manipulation planning for deformable linear objects. *IEEE Transaction on Robotics*, 23:1141–1150.

[Shapiro et al., 2007] Shapiro, A., Kallmann, M., and Faloutsos, P. (2007). Interactive motion correction and object manipulation. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 137–144, New York, NY, USA. ACM.

[Shi et al., 2007] Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H., and Guo, B. (2007). Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Transactions on Graphics*, 26(3):81.

[Shin et al., 2003] Shin, H. J., Kovar, L., and Gleicher, M. (2003). Physical touch-up of human motions. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 194–203, Washington, DC, USA. IEEE Computer Society.

[Shin and Oh, 2006] Shin, H. J. and Oh, H. S. (2006). Fat graphs: constructing an interactive character with continuous controls. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 291–298, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Shum et al., 2008a] Shum, H. P. H., Komura, T., Shiraishi, M., and Yamazaki, S. (2008a). Interaction patches for multi-character animation. *ACM Transactions on Graphics*, 27(5):1–8.

[Shum et al., 2007] Shum, H. P. H., Komura, T., and Yamazaki, S. (2007). Simulating competitive interactions using singly captured motions. *Proceedings of ACM Virtual Reality Software Technology 2007*, pages 65–72.

[Shum et al., 2008b] Shum, H. P. H., Komura, T., and Yamazaki, S. (2008b). Simulating interactions of avatars in high dimensional state space. *ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D) 2008*, pages 131–138.

[Si and Gaertner, 2005] Si, H. and Gaertner, K. (2005). Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proc of the 14th International Meshing Roundtable*, pages 147–163.

[Smith, 2007] Smith, R. (2007). Open dynamics engine. http://www.ode.org/.

[Sorkine et al., 2004] Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188.

[Stewart and Trinkle, 1996] Stewart, D. and Trinkle, J. C. (1996). An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691.

[Sumner and Popovic, 2004] Sumner, R. W. and Popovic, J. (2004). Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):397–403.

[Sutton and Barto, 2005] Sutton, R. S. and Barto, A. G. (2005). *Reinforcement Learning: An Introduction*. A Bradford Book, The MIT Press.

[Tak et al., 2000] Tak, S., young Song, O., and Ko, H.-S. (2000). Motion balance filtering. *Computer Graphics Forum*, 19(3):437–446.

[Takamatsu et al., 2006] Takamatsu, J., Morita, T., Ogawara, K., Kimura, H., and Ikeuchi, K. (2006). Representation for knot-tying tasks,. *IEEE Transactions on Robotics*, 22(1):65–78.

[Treuille et al., 2007] Treuille, A., Lee, Y., and Popovic', Z. (2007). Near-optimal character animation with continuous control. *ACM Transactions on Graphics*, 26(3):7:1–7:7.

[Van De Panne and Lamouret, 1995] Van De Panne, M. and Lamouret, A. (1995). Guided optimization for balanced locomotion. In Terzopoulos, D. and Thalmann, D., editors, *6th Eurographics Workshop on Animation and Simulation, Computer Animation and Simulation, September, 1995*, Eurographics, pages 165–177, Maastricht, Pays-Bas. Springer.

[Wakamatsu et al., 2006] Wakamatsu, H., Arai, E., and Hirai, S. (2006). Knotting/unknotting manipulation of deformable linear objects. *International Journal of Robotics Research*, 25(4):371–395.

[Wang et al., 2008] Wang, J. M., Fleet, D. J., and Hertzmann, A. (2008). Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, pages 283–298.

[Wang et al., 2009] Wang, J. M., Fleet, D. J., and Hertzmann, A. (2009). Optimizing walking controllers. *ACM Transactions on Graphics*, 28(5):1–8.

[Witkin and Kass, 1988] Witkin, A. and Kass, M. (1988). Spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 88)*, 22:159–168.

[Wooten and Hodgins, 1996] Wooten, W. L. and Hodgins, J. K. (1996). Animation of human diving. *Computer Graphics Forum*, 15(1):3–13.

[Xu et al., 2007] Xu, W., Zhou, K., Yu, Y., Tan, Q., Peng, Q., and Guo, B. (2007). Gradient domain editing of deforming mesh sequences. *ACM Transactions on Graphics*, 26(3):84.

[Yamane et al., 2004] Yamane, K., Kuffner, J. J., and Hodgins, J. K. (2004). Synthesizing animations of human manipulation tasks. *ACM Transactions on Graphics*, 23(3):532–539.

[Yamane and Nakamura, 2003] Yamane, K. and Nakamura, Y. (2003). Dynamics filter - concept and implementation of online motion generator for human figures. *IEEE Transactions on Robotics*, 19(3):421–432.

[Ye and Liu, 2010] Ye, Y. and Liu, C. K. (2010). Synthesis of responsive motion using a dynamic model. *Computer Graphics Forum*, 29(2):555–562.

[Yin et al., 2008] Yin, K., Coros, S., Beaudoin, P., and van de Panne, M. (2008). Continuation methods for adapting simulated skills. *ACM Transactions on Graphics*, 27:Article 81.

[Yin et al., 2007] Yin, K., Loken, K., and van de Panne, M. (2007). Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):Article 105.

[Zayer et al., 2005] Zayer, R., Rössl, C., Karni, Z., and Seidel, H.-P. (2005). Harmonic guidance for surface deformation. *Computer Graphics Forum*, 24(3):601–609.

[Zhou et al., 2005] Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., and Shum, H.-Y. (2005). Large mesh deformation using the volumetric graph laplacian. *ACM Transactions on Graphics*, 24(3):496–503.

[Zhou et al., 2010] Zhou, K., Xu, W., Tong, Y., and Desbrun, M. (2010). Deformation transfer to multi-component objects. *Computer Graphics Forum (Proceedings of Eurographics 2010)*, 29(2):319–325.

[Zordan and Hodgins, 2002] Zordan, V. B. and Hodgins, J. K. (2002). Motion capture-driven simulations that hit and react. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 89 – 96.

[Zordan et al., 2005] Zordan, V. B., Majkowska, A., Chiu, B., and Fast, M. (2005). Dynamic response for motion capture animation. *ACM Transactions on Graphics*, 24(3):697–701.